PROBLEMAS PROPUESTOS DE PROGRAMACIÓN

Con los problemas que se enuncian a continuación se busca que el estudiante posea una amplia muestra de propuestas a ser resueltas por medio del computador, elaborando un programa. Se debe hacer énfasis en la lógica de solución sin importar el lenguaje que se desee emplear y se deja, en términos generales, a su criterio la forma en como se comunicará con el usuario.

También es recomendable la generación de funciones y/o procedimientos para plantear las soluciones, en buena medida para fomentar la programación modular, pero como se notará muchos de los problemas planteados pueden ser resueltos como un simple programa, esta decisión vendrá en función de las necesidades puntuales que tenga el estudiante en un momento particular de sus estudios.

Se recomienda que se investigue los diferentes algoritmos existentes para resolver el problema planteado, ya que en algunos casos se explican brevemente, o tan sólo se mencionan.

Recuerde que su responsabilidad como programador implica que debe evitar todo tipo de error que se pueda presentar en una codificación (sintaxis, ejecución y lógica), como estrategia se le recomienda resolver primero lo que corresponde al problema planteado, y luego empezar a ubicar cuáles son los posibles obstáculos de funcionamiento que se puedan tener.

Así mismo es de resaltar que el estudiante puede plantearse variantes en los problemas planteados para que la entrada y salida de datos sea también a través de archivos tipo texto, lo que permitirá ejercitar esta área. Lo que si debe de cumplirse, es que todo problema propuesto implica que siempre se debe presentar el (o los) resultado(s).

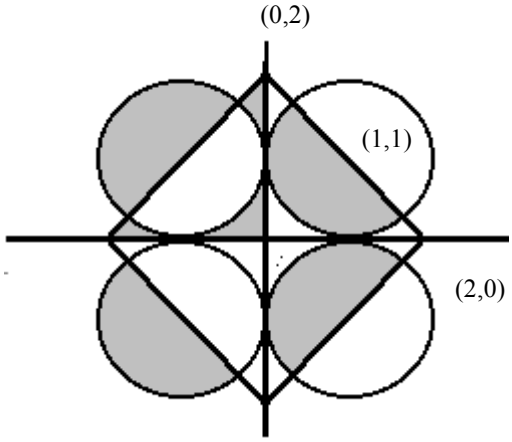
Es posible que en algunos problemas no se especifique de manera explícita la validación de datos, pero el estudiante debe partir en el planteamiento de sus soluciones siempre con la premisa de verificar todas las condiciones, que deben cumplir los datos que el usuario deba ingresar al programa y tomar las acciones pertinentes para cada caso. En este aspecto hay que hacer énfasis en que es una pésima práctica dejar en el usuario la responsabilidad de validar previamente los datos que vaya a introducir, para el programador puede ser más cómo pero el producto final es deficiente y alejado de las buenas prácticas en el desarrollo de programas.

PROBLEMAS INICIALES

1. Diseñe un programa que dadas las coordenadas (x,y) de un punto en el plano, determine y muestre sus coordenadas polares (R,θ).
2. Diseñe un programa que dados dos puntos en el plano por sus coordenadas (x_1, y_1) y (x_2, y_2) calcule y muestre la longitud del segmento que determinan estos puntos, y que se calculen y muestren las coordenadas del punto medio de ese segmento.
3. Dados dos puntos en el plano por sus coordenadas (x_1, y_1) y (x_2, y_2), diseñe un programa para calcular la ordenada correspondiente a una abscisa x cualquiera (también suministrada como dato) empleando **interpolación lineal**.
4. Diseñe un programa que dada una cantidad de segundos, entera y positiva, indique a cuanto equivale en años, meses, días, horas y segundos. Asuma años de 360 días y simplifique a que todos los meses poseen 30 días. Por ejemplo: 31.803.310 segundos equivalen a 1 año, 3 días, 2 horas, 15 minutos y 10 segundos.

5. Diseñe un programa que, dado un recipiente cilíndrico, solicite los datos pertinentes para poder calcular su **volumen** y **área**, y mostrar tales resultados.
6. Diseñar un programa que dadas las coordenadas **cartesianas** (*x*, *y*, *z*) de un punto en el espacio, calcule y muestre sus coordenadas **cilíndricas** y sus coordenadas **esféricas**.
7. En un reloj de agujas (que sólo tiene horario y minutero), elabore un programa que calcule y muestre el **menor ángulo** en grados que forman tales agujas dada una hora suministrada por el usuario.
8. Diseñe un programa que dado un punto en el plano por sus coordenadas (*x*,*y*), determinar en qué **cuadrante** se encuentra indicándolo con un mensaje al usuario.
9. Dados **tres** puntos en el plano por sus coordenadas (*x, y*), realice un programa que indique si los mismos se encuentran sobre una misma recta (si son **Puntos Colineales**).
10. Diseñe un programa que indique si tres puntos dados por sus coordenadas (*x*, *y*) **forman triángulo** y en caso de formarlo debe indicarse que **tipo** de triángulo forma.
11. Dadas tres **longitudes** suministradas por el usuario, diseñe un programa que indique si las mismas pueden **formar triángulo** y de ser así también calcule y muestre el área del triángulo.
12. Dados cuatro valores numéricos, leídos como datos, realice un programa que indique si los mismos pueden **formar un rectángulo**, en cuyo caso también se debe calcular y mostrar el área del mismo.
13. Una forma de determinar si un año es **bisiesto** es que el mismo sea divisible de manera exacta por cuatro, pero si el mismo es fin de siglo (año secular) en este caso debe ser divisible de forma exacta por cuatrocientos. Realice un programa que basado en el criterio anterior determine si un año dado como dato es o no bisiesto, generando un mensaje adecuado.
14. Ork el planeta natal de Mork celebra un gran festejo planetario cada ocho años, de manera similar cada 72 años se celebra el cumpleaños de Orson y para hacerlo en grande se festeja también al año siguiente, pero cada 48 años la celebración que pudiese haber ese año se suspende debido al penoso recuerdo que dejó la derrota con su planeta enemigo en las Guerras Impúdicas. Realice un programa para saber si un año dado como dato es o no festivo. Asuma que el conteo de los años empezó en el año uno.
15. Diseñar un programa que al introducir parte de la fecha de nacimiento (día y mes) muestre el nombre del **signo del zodíaco** correspondiente. Asuma las fechas estándar ofrecidas para cada signo.
16. Las siguientes unidades de distancia son de origen hispanoamericano y se muestra también su equivalencia con el metro: Almud (0,27 m), Cana (1,541 m), Dedo (0,0174 m), Estadal (3,391 m), Jarocha (4,19 m), Palmo (0,212 m), Mecate (20,062 m). Realice un programa donde dada una medida suministrada en metros, se indique su **equivalente** en la medida seleccionada por el usuario.
17. Realizar un programa que determine el mayor, el menor y el valor intermedio de **tres** números enteros y positivos dados como datos. Considere que el usuario puede suministrar los valores en cualquier orden.
18. Hacer un programa que, dada una **hora en el espejo**, devuelva la hora real. Por ejemplo, las 8:05 vista en el espejo es las 3:55.

19. Diseñe un programa que, dada la longitud de tres segmentos de recta, indique si los con ellos se puede formar un triángulo **Acutángulo**, **Rectángulo** o **Obtusángulo**.
20. Realice un programa que dados cuatro valores **A**, **B**, **C**, **D** los presente, en pantalla, ordenados de menor a mayor. Considere que el usuario puede suministrar los valores en cualquier orden.
21. Dados los datos que permitan formar la ecuación de segundo grado $\mathbf{Ax^2 + Bx + C = 0}$, genere un programa que calcule las **raíces** de dicha ecuación; contemple la posibilidad de mostrar resultados **imaginarios**.
22. Dados los conjuntos de puntos pertenecientes al círculo cuya ecuación es $\mathbf{x^2+y^2 =16}$, a la elipse determinada por la ecuación $\mathbf{x^2/36+y^2/16 =1}$ y a la recta cuya ecuación es $\mathbf{y =2x+1}$ diseñar un programa que para un punto definido por sus coordenadas (*x, y*), leídos como datos, se indique la figura (o conjunto de ellas) a las que pertenece.
23. Realice un programa que determine, en una sola instrucción de decisión, si un punto de coordenadas (*x, y*) cae dentro del área sombreada, todos los círculos son de radio igual a uno.



24. Realice un programa que permita al usuario llevar adelante una de las siguientes operaciones para **números imaginarios**: suma, resta, multiplicación o división. El usuario debe poder seleccionar una de las operaciones, dar los valores y obtener el resultado.
25. En un tablero de ajedrez se coloca en la primera casilla un grano de arroz, en la segunda se coloca el doble de granos que en la anterior y así sucesivamente hasta la $64^{\mathbf{ava}}$ casilla. Diseñe un programa para presentar un listado en donde se indique el número de la casilla, el número de granos colocados en esa casilla y la suma de los granos acumulados hasta esa casilla.
26. Desarrolle un programa que indique si un número entero positivo dado como dato es lo que se conoce como número **primo** o no.
27. Dado un número entero y positivo, genere un programa que **invierta los dígitos** del mismo; esto es: dado el número 1234 se debe obtener el número 4321. Como condiciones se debe tomar en cuenta que no se debe pedir el número de dígitos al usuario y tampoco se debe emplear ninguna función o procedimiento para trabajar con valores alfanuméricos.

28. Conocidas las coordenadas (*x, y*) de todos los vértices de un triángulo diseñe un programa que indique cuantos puntos de **coordenadas enteras** se encuentran **dentro** de tal triángulo.

29. Diseñe un programa que indique si un número entero y positivo "**M"** leído como dato, es ***perfecto*** o no. Sabiendo que un número es perfecto cuando la suma de todos sus divisores, salvo él mismo, es igual al mismo número. Por ejemplo: 6 es dividido de manera exacta por 1, 2 y 3 que al sumarlos da el valor origina: 6.

30. Un **número capicúa** es aquel que leído de izquierda a derecha es el mismo que leído de derecha a izquierda, por ejemplo 124421. Desarrolle un programa que determine si un número dado por el usuario es capicúa o no. Como condición se le impone que no puede usar artificios basados en el manejo de cadenas alfanuméricas.

31. Un número entero positivo se le llama **primo capicúa** cuando siendo primo también es primo cuando se invierten sus dígitos. Así tenemos que 17, 31, 37 y 113 son ejemplos de primos capicúas pues 71, 13, 73, 311 son también primos. Diseñe un programa para indicar si un número **N** dado como dato es primo capicúa.

32. Se dice que dos números **M** y **N**, enteros y positivos, profesan ***Amistad Matemática*** entre sí, cuando al sumar los divisores de **M** (salvo él mismo) se obtiene **N** y al sumar todos los divisores de **N** (salvo él mismo) se obtiene **M**. Por ejemplo:

    - los divisores de **220** son: 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 que al sumarlos da 284, y
    - los divisores de **284** son: 1, 2, 4, 71, 142 que al sumarlos da 220

    Generar un programa que dados dos números indique si los mismos profesan mutua amistad matemática.

33. Existe lo que se llama la ***Amistad Cuadrática*** entre dos números cuando se cumple lo que se narra en el siguiente ejemplo para los números 13 y 16:

    El número **16** elevado al cuadrado da 256 y sumando sus dígitos 2+5+6-->13

    El número **13** elevado al cuadrado da 169 y sumando sus dígitos 1+6+9--> 16

    Cuando lo anterior sucede se dice que los números profesan amistad cuadrática. Realizar un programa que indique si dos números dados como datos profesan amistad cuadrática.

34. Realice un programa que calcule y muestre el **Máximo Común Divisor** de dos números naturales dados como datos.

35. Desarrolle un programa que dados dos números naturales **N** y **M** suministrados por el usuario, calcule el **mínimo común múltiplo** (mcm) de los mismos.

36. En la filosofía de ¿Por qué hacerlo fácil? ¡Si se puede hacer difícil! Debe saber que la parte entera de la raíz cuadrada de un número, puede calcularse como la cantidad de sumas de números impares sucesivos que hay que hacer para totalizar el número sin sobrepasarse del mismo. Como ejemplo tome el número 36 y obtiene 1+3+5+7+9+11 cuya suma es 36 observe que se han tomado 6 valores y esta cantidad es la raíz cuadrada de 36 (se le invita a hacer la prueba con otros valores). Realice un programa en que basado en el criterio anterior calcule la parte entera de la raíz cuadrada de un número entero positivo **N**, leído como dato.

37. Un número natural se define como **Automórfico** cuando su cuadrado tiene como últimas cifras las mismas que ese número. Los primeros números automórficos son 5, 6, 25, 76, 376, 625... En efecto: 5^2=25, 6^2=36, 25^2= 625, 76^2= 5776.

38. Definamos **N** como **número suficiente** si cumple las siguientes características: ser entero positivo, impar y al sumar todos sus divisores (salvo él mismo) da un valor superior a **N**. Diseñe un programa que indique cuántos números suficientes existen por debajo de un valor **M** dado como dato.

39. Un **número narcisista** es aquel número entero y positivo, que es igual a la suma de cada uno de sus dígitos elevados a la "n" potencia (donde "n" es el número de cifras del número). La metáfora de su nombre alude a lo mucho que parecen "quererse a sí mismos" estas cifras. Por ejemplo, el **153** es un número narcisista puesto que $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$. Los primeros números narcisistas son: 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474 y 54748. Realice un programa que indique si un número entero y positivo leído como dato es un número narcisista.

40. Se dice que un número natural es **Odioso** cuando su equivalente en notación binaria presenta una cantidad impar del número 1. Realice un programa que dado un número entero suministrado por usuario indique si cumple con la definición ofrecida.

41. Se dice que un número natural es **Poderoso** cuando al menos uno de sus divisores (excepto el 1) es primo y que este divisor elevado al cuadrado, también divida de forma exacta al número. Por ejemplo: el número 36 tiene los siguientes divisores 2, 3, 4, 6, 9 y 12 donde 2^2 es 4, que divide de forma exacta a 36 (igual pasa con 3). Realice un programa que dado un número entero suministrado por usuario indique si cumple con la definición ofrecida.

42. Se dice que un número natural es **oblongo**: si cumple con ser el producto de dos naturales consecutivos. Por ejemplo, los números 30, 42 y 56 lo son. Diseñe un programa que indique si un número dado por el usuario cumple con la condición señalada.

43. Una **pareja prima** es un par de números primos separados por dos unidades (como ejemplo tome 11 y 13, así como el 29 y el 31). Realice un programa para contabilizar cuantas parejas primas se encuentran entre 2 y un valor **N** dado como dato por el usuario.

44. Se dice que un número natura es **factorion** si al sumar los factoriales de los dígitos que lo componen, dan el mismo número. Por ejemplo, para el número 145 tenemos: 1! + 4! + 5! = 145. Realice un programa que indique si un número entero y positivo dado como dato por el usuario cumple con esta particular característica. Nota: 1! = 1 y 2! = 2 no son sumas y por lo tanto no se deben incluir. Asuma no disponer la función factorial, ni debe emplear funciones de manejo de alfanuméricos.

45. Se dice que un número entero y positivo de N dígitos es **n-pandigital** si dicho número contiene todos los dígitos de 1 a N sólo una vez. Por ejemplo, los números 15423 y 12435 son 5-pandigitales. Realice un programa que dado un valor leído como dato indique si el mismo es **n-pandigital.** Como limitante, no puede emplear funciones para manejos de alfanuméricos.

46. Se dice que tres números naturales **A**, **B** y **C** forman ***terna pitagórica*** cuando se cumple la relación $\mathbf{A^2 + B^2 = C^2}$. Por ejemplo 3, 4 y 5 forman una terna pitagórica ya que $3^2 + 4^2 = 5^2$. Diseñe un programa que cuente cuántas ternas pitagóricas se forman tales que tanto el Valor de **A** y de **B** sean menores que un número **N** dado como dato.

47. Se dice que un número entero y positivo es **cubifinito** cuando al elevar todos sus dígitos al cubo y sumarlos el resultado es 1, o da otro número cubinfinito. Por ejemplo, el número 1234 realizamos lo siguiente $1^3+2^3+3^3+4^3 = 100$ que es cubifinito. Como contraejemplo el número 513 no cumple con lo indicado. Elabore un programa que indique si un número dado por el usuario cumple con esta definición.

48. Diseñe un programa que encuentre el **menor** valor entero de **X** que hace que el binomio $\mathbf{p(x)=991x^2 + 1}$ sea un ***cuadrado perfecto***.

49. Realice un programa que dado un número real dado por el usuario calcule el número (cantidad) de cifras enteras y el número (cantidad) de cifras decimales que lo conforman. Como condición se le impone que no puede usar artificios basados en el manejo de cadenas alfanuméricas o funciones que realicen directamente la actividad solicitada.

50. Realice un programa que, dado un número en **base diez** (entero y positivo), calcule y presente su equivalente en **base binaria**, o en **base octal**. Pida al usuario que tipo de conversión desea.

51. Realice un programa que, dado un número en **base binaria**, calcule y presente su equivalente en **base diez**.

52. Note que el número 58510 (base decimal) es equivalente a 1001001001 (base binaria). También debe observar que dicho número es capicúa en ambas bases. Diseñe un programa que indique si un número natural dado por el usuario cumple con esta característica.

53. Realice un programa que calcule la **suma** de todos los dígitos de los números del 1 a un número **N** suministrado por el usuario.

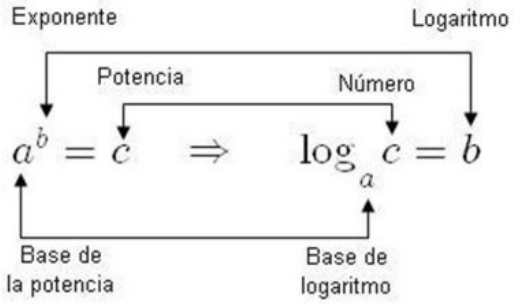
54. Hay *geeks* que afirman que el mejor número es el 73 y el razonamiento para tal declaración se sustenta en varios hechos curiosos: a) 73 es el 21^{avo} número primo, b) Su espejo, el número 37, es el 12^{vo} número primo, c) 21 es el producto de multiplicar 7 por 3, y finalmente c) en su representación binaria, 73 es un capicúa: 1001001.

    Escriba programas que le permitan responder las siguientes preguntas:

    - ¿Existen otros valores *p* que sean el *n*-ésimo primo, tales que espejo(*p*) es el espejo(*n*)-ésimo primo?
    - ¿Existen otros valores *p* que sean el *n*-ésimo primo, tales que *n* es el producto de los dígitos de *p*?
    - ¿Cuáles son los primeros diez números primos cuya representación binaria es un capicúa?

55. El algoritmo de Euclides para encontrar el máximo común divisor (***mcd***) de dos números es el siguiente: Dados los enteros **A** y **B** (donde A>B), se divide A entre B obteniendo el cociente Q1 y el resto R1. Si R1 es diferente de cero, se divide B por R1, obteniendo el cociente Q2 y el resto R2. Si R2 es distinto de cero, se divide R1 por R2, obteniendo cocientes y restos sucesivos. el proceso continúa hasta obtener un resto igual a cero. El resto anterior a este es le **mcd** de A y B. Diseñe un programa que calcule el **mcd** de dos valores dados como datos mediante el algoritmo de Euclides.

56. Diseñe un programa que calcule el **logaritmo en base diez** de un número entero suministrado como dato, asuma que no tiene disponible en el lenguaje que está empleando la función para calcular directamente dicho algoritmo. Como ayuda vea la figura en la que se le recuerda el origen del algoritmo.



57. Se dice que un número natural es **Ambicioso** si cumple que si sumamos sus divisores propios (salvo el mismo número), repitiendo este proceso varias veces, acabamos obteniendo un número perfecto. Por ejemplo, para 95 es un número ambicioso, porque sus divisores son 1, 5, 19 y tenemos que 1+5+19=25, y los divisores de 25 son 1 y 5, por lo que 1+5=6, y 6 es un número perfecto. Realice un programa que indique si un número dado por el usuario cumple con esta definición.

58. Diseñe un programa que determine si un número **N** entero y positivo, leído como dato, es o no un **número primo**, sin emplear ni la multiplicación, ni la división en ninguna de sus variantes, ni una función o procedimiento estándar del lenguaje en el que se trabaje.

59. Se dice que un número natural (**N**) es **Número de Mesenne** si siendo un número primo, se cumple que 2^n-1 también es un número primo. Realice un programa que dado un número entero suministrado por usuario indique si cumple con la definición ofrecida. Ejemplos de este número son: 3, 7, 31, 127, 8191… resaltando no son muchos los encontrados hasta el momento.

60. En 1973, el matemático inglés Neil Sloane definió, definió en una revista dedicada a las matemáticas recreativas, la **persistencia multiplicativa** de los números. Consiste en el número de veces que hay que multiplicar los dígitos de un número (escrito en base 10) hasta llegar a un número de un único dígito.

    Por ejemplo, el número 39 tiene una persistencia multiplicativa de 3: 39 →3*9=27→2*7=14→1*4=4

    Dada la definición anterior diseñe un programa que, dado un número natural por parte del usuario, calcule su persistencia multiplicativa.

61. Los ***números cadena*** son creados al sumar continuamente el cuadrado de cada dígito de un número para formar otro. Por ejemplo, si se tiene el número 44, se procede a elevar cada uno de sus dígitos al cuadrado y la suma de ellos resulta en 32 ($4^2 + 4^2 = 16 + 16 = 32$). Luego se hace lo mismo con el 32 ($3^2 + 2^2 = 13$) y así sucesivamente. Por ejemplo:

    44 →32 →13 →10 →**1** →**1**
    85 →**89** →145 →42 →20 →4 →16 →37 →58 →**89**

    Una vez que la cadena llegue a 1 u 89, la cadena queda en un ciclo infinito como los mostrados en los ejemplos anteriores. Es interesante notar que al comenzar con CUALQUIER número entero positivo, eventualmente llegará a 1 (y se le llama número Feliz) u 89 (llamado número infeliz).

    Se pide que diseñe un programa que permita conocer cuál es el porcentaje de números enteros menor a un valor **N** (leído como dato) cuya cadena llega al número 89.

62. Una forma de generar ***números aleatorios*** es la llamada técnica del cuadrado medio, que consiste en tomar un número de **N** cifras, elevarlo al cuadrado, y usar los **N** dígitos del medio como el siguiente elemento (semilla) al que se le aplicará la misma operación. Diseñe un programa que solicite a un usuario el valor de N para realizar este proceso, verificando que tal valor posea al menos dos dígitos. Como condición se le impone que no puede usar artificios basados en el manejo de cadenas alfanuméricas.

63. Realice un programa que calcule y muestre cuántos números enteros no tienen cifras repetidas dentro de un intervalo [**A**, **B**] dado como dato.

64. Un caracol cae en un pozo de **H** metros de profundidad. Este caracol durante el día asciende una distancia **Ld** metros, pero durante la noche, al quedarse dormido, resbala y desciende **Ln** metros. Diseñe un programa que simulando el movimiento del animalito, determine en cuánto tiempo sale del pozo (si es que esto sucede).

65. Una pelota cae verticalmente desde una altura de **H** metros y rebota cada vez hasta un 32% de su altura previa. Diseñe un programa que determine la distancia vertical total que recorre la pelota antes de quedar en reposo. Se considerará reposo un rebote menor que 10^{-4} metros.

66. Se deja caer una pelota desde una altura **H1** hacia una superficie vertical totalmente plana, en cada rebote la altura máxima del mismo viene dada por la pérdida de un porcentaje **P** de la máxima altura alcanzada en el rebote previo (en el primer caso se toma la altura desde donde se le deja caer). Realice un programa que simule el proceso indicado y señale luego de cuántos rebotes la altura máxima no llega a una altura **H2** (obviamente **H2** < **H1** y que debe ser verificado).

67. Una persona planea ahorrar una cantidad fija **C** por mes a una tasa de interés **I** anual que capitalice mensualmente. Realice un programa que determine el número de años y meses requeridos para acumular, al menos, una cantidad final **TOTAL** si los intereses son abonados mensualmente.

68. Una persona planea ahorrar una cantidad fija **C** por mes a una tasa de interés **I** anual que capitalice mensualmente, suponga que la persona al finalizar cada año, justo después de sumar los intereses, hace un retiro por una cantidad total de ***Y*** bolívares. Diseñe un programa que determine cuál será el primer año en que su balance no permita que haga semejante retiro.

69. Aquiles, el famoso guerrero griego, entabla una carrera con una tortuga a la cual le da ***V*** metros de ventaja. La tortuga se desplaza a una velocidad ***Vtor*** metros por hora y Aquiles es **diez** veces más veloz que ella. Diseñe un programa que simule la carrera y genere como resultados el tiempo en horas, minutos y segundos que tarda Aquiles en alcanzar a la tortuga, y la distancia recorrida desde el punto de partida al punto donde le da alcance.

70. Una persona se entera de un secreto, al cabo de un mes no se contiene y se lo cuenta a otras tres personas, pero luego no lo vuelve a comentar. Cada una de estas tres personas tienen un comportamiento similar al conservarlo durante un mes, al cabo del cual lo cuentan a otras tres personas y luego no lo vuelven a contar. Suponiendo que nunca le cuentan este supuesto secreto dos veces a la misma persona, y que se mantiene este proceso de propagación; diseñe un programa que calcule en cuántos meses se entera una población de **N** habitantes.

71. Sabiendo que la población en Venezuela en 1984 fue de 16 millones de habitantes, y **A** es la tasa promedio de crecimiento anual esperada (que incluye los nacimientos e inmigraciones, menos los decesos y emigraciones), realice un

programa que estime la población anual esperada hasta un futuro año **X**. La tasa **A** está expresada en porcentaje y el programa debe simular el proceso de crecimiento de la población.

72. Sabiendo que una pareja de conejos da a luz a otra pareja cada mes a partir de los dos meses de vida, realice un programa que determine cuántas parejas habrá al cabo de **N** meses si partimos de una pareja recién nacida. Considere nula la mortalidad de los conejos en estos **N** meses.

73. Se tiene un reloj que atrasa un minuto por cada cuarto de hora que transcurre en tiempo real. Se sincroniza este reloj con otro que funciona correctamente a la medianoche del primer día de un experimento. Diseñe un programa que permita conocer qué hora marcaría el reloj defectuoso una vez transcurridos **D** días y **H** horas desde el inicio del experimento en cuestión, considere como condición que el programa debe simular la situación planteada.

74. Un tanque de agua de **capacidad** Q (litros) es llenado de manera intermitente por un surtidor de agua que es activado cada T segundos durante T segundos, y la cantidad de agua que este surtidor aporta viene dado por V surtido = T*V, donde V es el volumen en litros por unidad de segundo que suministra. Dicho tanque tiene un desagüe en la parte inferior que pierde agua a razón de B (litros / segundo). Realice un programa que **simule** la situación segundo a segundo y que indique el estado del tanque al cabo de N segundos. Asuma que el tanque inicialmente contiene I litros (siempre y cuando $0 < I < Q$).

75. En una planta de tratamiento una población de bacterias limpiadoras es utilizada para eliminar las bacterias contaminantes del agua, las bacterias contaminantes se duplican cada 5 segundos, pero las bacterias limpiadoras se comen a las bacterias contaminantes cada segundo y luego de haber engullido 3 la bacteria limpiadora se duplica y muere de una manera inexplicable. En un estanque de capacidad **Q** (litros, $1.000 \leq \mathbf{Q} \geq 10.000$), hay **M** bacterias contaminantes por cada litro de agua. Se desea saber la cantidad mínima de litros (en un valor entero), de producto limpiador que hay que suministrar en el estanque para poder eliminar todas las bacterias contaminantes, sabiendo que hay $1x10^7$ bacterias limpiadoras por litro de producto limpiador. Resuelva este problema mediante un programa que simule el proceso de nacimiento y muerte de las bacterias.

76. Dos colonias de bacterias **A** y **B**, comparten un cultivo de laboratorio. Los investigadores han hecho las siguientes observaciones:

- Ambos tipos de bacterias se reproducen por división celular, en dos bacterias nacientes. Las bacterias **A** se reproducen cada dos minutos y las **B** cada minuto.
- Cada bacteria naciente **A** devora:
  - 3 bacterias nacientes **B**, si la población **A** es menor o igual que la tercera parte de la población **B**.
  - 2 bacterias nacientes **B**, si la población **A** es mayor que la tercera parte de la población B, pero menor o igual que la mitad de la misma.
  - 1 bacteria naciente **B**, si la población **A** es mayor que la mitad de la población **B**.
- A los 10 minutos de iniciado el estudio y sucesivamente cada 10 minutos, por causas inexplicables, después de devorar a su(s) presa(s), muere la mitad de la población **A**.

Analice el problema y diseñe un programa que, conocidas las poblaciones **A** y **B** en un instante cualquiera, determine las poblaciones de ambas colonias al cabo

de **N** minutos. Determine además si hay extinción de alguna colonia en el lapso estudiado.

Consideraciones:

- No hay otros motivos de mortalidad distintos de los mencionados.
- En el instante inicial del estudio todas las bacterias son recién nacidas.
- Una bacteria **A** tarda menos de 1 minuto en devorar a su(s) presa(s).

77. Realizar un programa que calcule el **factorial** de un número entero y positivo, no olvide tomar las precauciones pertinentes para cubrir el eventual problema de sobrepasar el rango de valores numéricos que dispone en sus variables.

78. Diseñar un programa que calcule la **división de factoriales** de dos números naturales suministrados por el usuario. Trate de cubrir la mayor cantidad de valores sin sobrepasar las capacidades del computador.

79. Diseñe un programa que le permita calcular mostrar el resultado la siguiente suma 1/1 + 1/2 + 1/3 + ... + 1/N, donde **N** es un número entero y positivo leído como dato.

80. La suma de los números primos menores a 10 es 2 + 3 + 5 + 7 = 17. Realice un programa que determine la **suma de los primos** menores a un valor **N** entero y positivo leído como dato.

81. Los Números de Fibonacci están formados por la siguiente sucesión:

1, 1, 2, 3, 5, 8, 13, 21, ….

Luego de que defina la ley de formación de esta serie, escriba un programa que presente en pantalla el ***n-simo*** valor de esta serie, donde **n** es un dato dado por el usuario, debe verificar cual es el valor máximo que puede ser empleado en el computador.

82. Realice un programa que lleve a cabo la siguiente suma para un valor de X, leído como dato, que cumpla con la condición (0 < X <1):

$$S = Ln(Ln(X)) + Ln(X) + \frac{Ln^2(X)}{2} + \frac{Ln^3(X)}{3} + \frac{Ln^4(X)}{4} + \ldots$$

Debe detener el cálculo cuando el último término generado sea menor, en valor absoluto, que un valor **EPS,** también leído como dato.

83. El valor de **PI** (π) puede ser calculado empleando las expresiones abajo indicadas. Realice un programa (o varios, según su gusto) que calcule el valor de π mediante el desarrollo de las series indicadas y compárelo con el valor de la función PI del lenguaje que emplee. Indique en cada caso, cuántos términos deben sumarse para producir un valor de π que difiera del valor dado por la función de biblioteca, en menos de un valor dado por el usuario, valor el cual debe, a su vez, ser menor que uno y mayor que cero:

a. $$1 + \frac{1}{3^4} + \frac{1}{5^4} + \frac{1}{7^4} + \ldots\ldots = \frac{\pi^4}{96}$$

b. $$\frac{1}{2} - \frac{1}{5} + \frac{1}{8} - \frac{1}{11} + \frac{1}{14} - \ldots\ldots = \frac{\pi\sqrt{3}}{9} - \tfrac{1}{3}ln\,2$$

c. $$\frac{1}{1^6} - \frac{1}{2^6} + \frac{1}{3^6} - \frac{1}{4^6} + \ldots\ldots = \frac{31\pi^6}{30240}$$

d. $$\frac{1}{1^2 2^2 3^2} + \frac{1}{2^2 3^2 4^2} + \frac{1}{3^2 4^2 5^2} + \ldots\ldots = \frac{4\pi^2 - 39}{16}$$

e. $1 + \frac{1}{3^3} - \frac{1}{5^3} - \frac{1}{7^3} + ... = \frac{\pi^3\sqrt{2}}{16}$

84. Dados dos números enteros y positivos, **M** y **K**, suministrados como datos por el usuario realizar un programa que calcule el **número combinatorio C** entre estos dos números, según la siguiente fórmula.

$$C = \binom{m}{k} = \frac{m!}{k!(m-k)!}$$

Para realizar este programa, asuma que no dispone de la función que calcula el factorial de un número.

85. Otra forma de generar los números de la Serie de Fibonacci es la que se muestra abajo (que se basa en el cálculo de valores combinatorios), genere un programa que muestre en n-esimo término de la serie.

$$F_{n+1} = \binom{n}{1} + \binom{n-1}{1} + \binom{n-2}{2} + ...$$

86. Realice un programa que genere la suma dada por la siguiente serie:

$$S = -X + \frac{X^2}{2!} - \frac{2X^3}{3!} + \frac{3X^4}{4!} - ...$$

Como condiciones debe verificar que se cumpla que el valor de **X** leído como dato sea con **0 < X < 1**. Obligatoriamente debe generar el término ***N-simo*** a partir del término anterior (en este caso como una unidad y no viéndolo como la generación del numerador y la del denominador por separado) y detenga el cálculo cuando la diferencia de dos términos consecutivos (en valor absoluto) sea menor que una tolerancia también dada por el usuario. La tolerancia también debe ser positiva y menor que uno.

87. Realice un programa que genere la suma dada por la siguiente serie:

$$S = (1-X)^2 - \frac{(1-X)^4}{3!} + \frac{(1-X)^6}{5!} - \frac{(1-X)^8}{7!} + ...$$

Como condiciones debe verificar que se cumpla que el valor de **X** leído como dato sea **0 < X < 1**. Obligatoriamente debe generar el término ***N-simo*** a partir del término anterior (en este caso como una unidad y no viéndolo como la generación del numerador y la del denominador por separado) y detenga el cálculo cuando la el último término, en valor absoluto, sumado sea menor que una tolerancia dada por el usuario. . La tolerancia también debe ser positiva y menor que uno.

88. Realice un programa que genere la suma dada por la siguiente serie:

$$S = \frac{(X-1)}{2!} + \frac{(X-1)^2}{3!} + \frac{(X-1)^3}{4!} + ... + \frac{(X-1)^{n-1}}{N!} + ...$$

Detener el cálculo de la suma cuando el último sumado sea menor en términos absolutos, a un valor de precisión, también leído como dato, previendo que si no converge en 20 iteraciones se imprima un mensaje que así lo señale y muestre el valor calculado hasta el momento.

89. Dada la serie:

$$\frac{\Pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots.$$

Diseñar un programa para calcular el valor de Π. Detenga el cálculo cuando la diferencia entre los dos **últimos términos sumados** en términos absolutos sea menor que un valor de precisión dado por el usuario. De forma adicional debe de contarse y mostrar el número de términos calculados.

90. El número **e** se puede calcular mediante la siguiente serie:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

Escribir un programa que calcule el valor del número ***e*** deteniendo el cálculo de la serie cuando el valor del último término generado sea menor que un valor de precisión dado por el usuario. Debe indicar igualmente cuántos términos fueron empleados en el cálculo.

91. El valor del coseno de un ángulo puede calcularse utilizando el siguiente desarrollo en ***serie de Taylor***:

$$Cos(X) = 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} + \frac{X^8}{8!} -$$

Diseñe un programa en que dado un ángulo en grados se genere su coseno de acuerdo al criterio anterior. Tome en cuenta que la serie desarrollada es válida para ángulos dados en radianes. Como condición genere cada término de la serie en función del anterior (entendiendo el término como un único valor y no el numerador y el denominador por separado). Detenga el cálculo de la serie cuando la diferencia entre los dos últimos sumados, en valor absoluto, sea menor que un valor de precisión dado por el usuario. Comparte este valor con el resultado de emplear la función preexistente en el leguaje que está empleando.

92. Dada una función continua y de signo constante sobre un intervalo [a, b] se le puede calcular el **área bajo la curva** aplicando la ***regla del rectángulo por el punto medio***, para esto se subdivide el intervalo [a, b] en **N** subintervalos de igual longitud (ancho = (b-a)/n)) y se aproxima el valor del área pedida mediante la suma del área de cada rectángulo definido por cada subintervalo donde la base es ancho y la altura se obtiene al evaluar la función en el punto medio del subintervalo. Diseñe un programa que calcule el área bajo la curva de una función, siguiendo las indicaciones anteriores.

93. Dada una función continua y de signo constante sobre un intervalo [a, b] se le puede calcular el **área bajo la curva** aplicando la ***regla trapecial***, para esto se subdivide el intervalo [a, b] en **N** subintervalos de igual longitud (ancho = (b-a)/n)) y se aproxima el valor del área pedida mediante la suma del área de cada trapecio definido por cada subintervalo donde la base es ancho y la altura se obtiene al evaluar la función en cada uno de los extremos del subintervalo. Diseñe un programa que calcule el área bajo la curva de una función, siguiendo las indicaciones anteriores.

94. Dado un ángulo expresado en grados, determinar el valor del **seno** del mismo utilizando el desarrollo en serie de ***Mac Laurin*** y comparar el valor contra el que proporciona la función que dispone en el lenguaje que emplea para el cálculo de tal función. El número de términos a sumar debe ser un dato pedido

al usuario, y recuerde pasar el ángulo a radianes para emplearlo en la serie. El desarrollo en serie es:

$$Sen(X) = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \ldots$$

95. El desarrollo de la **serie de Taylor** para $Y = l^X$, viene dado por la expresión:

$$l^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

Calcule el valor de serie para un valor **X** dado. Detenga el cálculo de la serie cuando la magnitud de la diferencia entre los **dos últimos términos sumados** sea menor que 0.0001. Adicionalmente deberá presentar el número de términos calculados. Como condición del problema, considere que no dispone de la función factorial.

96. Diseñe un programa que determine la cantidad de puntos de coordenadas (**x**, **y**) enteras que poseen como **Máximo Común Divisor** la unidad, esto es el MCD entre **x** y **y,** y que a su vez se encuentren dentro de la elipse $\mathbf{ax^2 + by^2 = r^2}$ (donde ***a*** y ***b*** deben ser valores positivos).

97. Una manera de calcular el **área de un triángulo** en base a sus lados es empleando la ***fórmula de Herón***:

$$Area = \sqrt{S(S-A)(S-B)(S-C)}$$

$$S = \frac{A+B+C}{2}$$

Donde **S** es el semiperímetro del triángulo y A, B y C son los lados del triángulo.

Realice un programa que solicite los tres vértices de un triángulo por sus coordenadas (x,y), verifique que tales puntos formen un triángulo y presente su área.

98. El Método de Newton para el cálculo de la **raíz N-ésima** de un número A ($A \in R$) procede de la siguiente manera:

- Se parte de una aproximación inicial Xant, generalmente inicializada en 1
- Se calcula la aproximación siguiente Xsig mediante la expresión:

$$Xsig = \frac{(n-1) * Xant + \frac{A}{Xant^{(n-1)}}}{n}$$

Si la magnitud de la diferencia entre Xant y Xsig es menor que una tolerancia **TOL**, se considera a Xsig como la raíz. En caso contrario, se toma el valor de Xsig como nuevo valor de Xant y se repite el paso anterior. Es un método abierto, en el sentido de que no está garantizada su convergencia global, lo cual debe tomar en cuenta en su código. La única manera de alcanzar la convergencia es seleccionar un valor inicial lo suficientemente cercano a la raíz buscada.

Diseñe un programa que dado un valor **A**, entero y positivo, y una tolerancia (**TOL**) mayor que cero y menor que uno, calcule y escriba el valor de la raíz **N-ésima** de **A** de una función, por el método anteriormente descrito.

99. La búsqueda (o método) de la **Sección Dorada** es una técnica para hallar el extremo (mínimo o máximo) de una función unimodal, mediante reducciones

sucesivas del rango de valores en el cual se conoce que se encuentra el extremo. Diseñar un programa que contenga una función que calcule el **máximo** de una función, por ejemplo: f(x) = e^x + sen(x) por el Método de la ***Sección Dorada***, dado un intervalo inicial de búsqueda (A,B).

Dicho método se basa en calcular dos puntos **L** y **M** interiores al intervalo (A,B) tales que:

$L = A + 0{,}382 * (B - A)$ y $M = A + 0{,}618 * (B - A)$

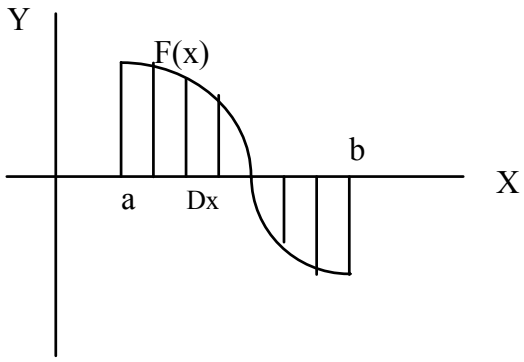
Si f(L) > f(M) entonces A = L; L = M y M = A + 0,618*(B-A)

en caso contrario B = M; M = L y L = A + 0,382*(B-A)

El proceso se repite hasta que (B-A) < ***Error***, donde ***Error*** es leído como dato.

100. La **longitud de arco de una curva**, también llamada rectificación de una curva, es la medida de la distancia o camino recorrido a lo largo de una curva o dimensión lineal. La longitud de una curva plana se puede aproximar al sumar pequeños segmentos de recta que se ajusten a la curva, esta aproximación será más ajustada entre más segmentos sean y a la vez sean lo más pequeño posible, escogiendo una familia finita de puntos, y aproximar la longitud mediante la longitud de la poligonal que pasa por dichos puntos. Cuantos más puntos escojamos, mejor sería el valor obtenido como aproximación de la longitud. Diseñe programa para hallar una buena aproximación de la **longitud de curva** de una función, por ejemplo: y = EXP(-X*X) en un intervalo [a,b] definido por el usuario.

101. Se tiene la función, por ejemplo: **F(x) = e^x + sen (x)** que cumple con ser continua en un intervalo a,b y en donde se estima existe una raíz única. Por tanto, en dicho intervalo los valores F(a) y F(b) tienen signos opuestos. Para aproximar el valor de la raíz puede aplicarse el **método de tanteos**, que busca el subintervalo posible donde se encuentra la raíz.



Para ello se define un valor de incremento **Dx = (b-a)/10**. Luego se procede a evaluar la función F(x) para valores de **x = a + Dx**, comparando los signos de F(a) y F(x). Si tienen el mismo signo, se incrementa **x** en **Dx** y se repite el proceso, hasta que en algún momento F(a) y F(x) tienen signos opuestos, lo cual indica que se pasó una raíz. Entonces el valor de **x** se disminuye en **Dx**, se hace **Dx = Dx/10** y se repite todo el proceso para los nuevos valores de **x** y **Dx**. El proceso repetitivo finaliza cuando **Dx** sea menor que un cierto valor **EPS** también leído como dato.

Realice un Programa que, utilizando funciones, determine la raíz de una función, por ejemplo: f(x) = e^x + sen(x) por el método antes descrito.

102. Otro método para poder encontrar la raíz de una función es el **Método de Bisección** es uno de los métodos más sencillos y de fácil intuición para resolver ecuaciones en una variable, también conocido como **Método de Intervalo Medio**.

El método consiste en lo siguiente:

- Debe existir seguridad sobre la continuidad de la función f(x) en el intervalo [a,b]

- A continuación se verifica que f (a) * f (b) < 0
- Se calcula **el punto medio** m del intervalo [a,b] y se evalúa f(m) si ese valor es igual a cero, ya hemos encontrado la raíz buscada
- En caso de que no lo sea, verificamos si f(m) tiene signo opuesto con f(a) o con f(b)
- Se redefine el intervalo [a, b] como [a, m] ó [m, b] según se haya determinado en cuál de estos intervalos ocurre un cambio de signo
- Con este nuevo intervalo se continúa sucesivamente encerrando la solución en un intervalo cada vez más pequeño, hasta alcanzar la precisión deseada

Diseñe un programa que calcule la raíz de una función sobre el método descrito.

103. Para calcular ***aproximaciones a la raíz de una función***, mediante métodos numéricos existen otros métodos a los antes mencionados, dos de ellos son el **Método de la Secante** y **Regula Falsi**, investigue su forma de aplicación y diseñe un programa que contenga funciones que los apliquen.

104. Escriba un programa que contenga una **función recursiva** que calcule el factorial de un número **N**, entero y positivo.

105. Escribir una **función recursiva** que determine el ***capital*** C_n obtenido, al situar un capital inicial C_0 a interés compuesto durante **N** años al interés anual **r** (expresado en porcentaje). Siendo la fórmula de interés compuesto:

$$C_n = C_0 * \left(1 + r/100\right)^n$$

CADENAS ALFANUMÉRICAS

En esta sección se presentan una serie de problemas que hacen especial énfasis en el manejo de cadenas alfanuméricas, se recomienda un adecuado repaso de las funciones y procedimientos que dispone para el trabajo con este tipo de datos en el lenguaje que está empleando.

En este punto es recomendable que replantee soluciones de problemas previos, ya que los mismos pueden ser resueltos con enfoque de uso de alfanuméricos y las funciones asociadas.

106. Por un momento ubíquese en los viejos tiempos trabajando para una compañía de telégrafos, ya que debe el calcular el costo del envío de un mensaje según los siguientes criterios: cada letra cuesta 10 unidades monetarias (u.m.), los caracteres especiales que no sean letras cuestan 30 u.m. y los dígitos tienen un valor de 20 u.m. cada uno, los espacios no tienen valor y las letras del castellano (ñ, á, é, í, ó, ú) se consideran caracteres especiales. Diseñe un programa que dada una cadena de caracteres indique el costo total del envío de tal mensaje.

107. Se dice que una palabra es **pentavocálica** cuando está dotada de las *cinco vocales sin repetición*, por ejemplo: murciélago, refugiado, abuelito, hipotenusa, cuadernillo, descuidado, educación, reputación, surrealismo, vestuario, euforia, y ultraligero. Diseñe un programa que dada una palabra leída como dato indique si la misma es **pentavocálica**.

108. ***Dos*** palabras son **anagramas** si tienen las mismas letras pero en otro orden. Por ejemplo, «torpes» y «postre» son anagramas, mientras que «aparta» y «raptar» no lo son, ya que «raptar» tiene una r de más y una a de menos. Diseñe un programa que indique si un par de palabras suministradas como datos son anagramas.

109. Imagina que usted es un biólogo que examina secuencias de ADN de formas de vida diferentes. Se le darán dos secuencias de ADN, y su objetivo es encontrar el conjunto ordenado de bases adyacentes de mayor tamaño que es común en ambos ADNs.

    Las secuencias de ADN se darán como conjuntos ordenados de bases de nucleótidos: adenina (abreviado A), citosina (C), guanina (G) y timina (T):

    Por ejemplo, sea la Cadena # 1: ATGTCTTCCTCGA y la Cadena #2: TGCTTCCTATGAC

    El resultado es CTTCCT porque que es el conjunto ordenado de bases adyacentes de mayor tamaño que se encuentra en ambas formas de vida. Desarrolle un programa que le permita llevar adelante sus investigaciones.

110. Diseñe un programa que determine el número de veces que aparece una palabra 'palabra' ingresada por un usuario en una cadena de caracteres también ingresada por el usuario.

111. Desarrolle un programa que lea una cantidad de palabras y calcule el largo de cada una, pero sin considerar las letras repetidas, determinando la más larga y más corta.

112. Desarrolle un programa que escriba en **números romanos** un número suministrado por el usuario en **base a diez**. Investigue como es la numeración romana y establezca el máximo valor que su programa podrá representar.

113. Diseñe un programa puede hacer la conversión de un número dado en guarismos **romanos** a su equivalente en **base a diez**.

114. Desarrolle un programa para jugar al popular juego **el ahorcado**, el cual consiste en un personaje, el cual está a punto de ser ejecutado. Para salvarlo es necesario adivinar una palabra, de la cual sólo se conoce su longitud. El jugador debe ir eligiendo letra por letra, de modo de ir completando la palabra. Si el jugador se equivoca en una letra, es decir, la letra seleccionada no se encuentra en la palabra a adivinar, se va dibujando parte de su cuerpo (un brazo, una pierna, el tronco, etc.) y la horca. Se puede jugar hasta que el personaje sea totalmente dibujado junto con la horca que acabará su trágica vida.

    Realice un programa que solicite a un jugador una palabra y que luego vaya solicitando letra a letra a la otra persona, el programa debe terminar cuando todas las letras de la palabra se hayan adivinado, es decir, ganar el juego, o bien se haya cometido un número establecido de desaciertos, es decir, gana el computador. Si desea incrementar la dificultad puede ir dibujando el muñequito con su horca por cada letra errada

115. Una compañía que promociona productos antiguos ha estado recibiendo telegramas de felicitación. El problema es que los telegramas se han enviado en código Morse y los espacios entre las letras se han perdido en la transmisión.

    Investigue las características de este código, pero es importante saber que en el código Morse, cada letra del alfabeto se sustituye por una secuencia de puntos y rayas de la siguiente manera:

a .-	h	o ---	v ...-
b -...	i ..	p .--.	w .--
c -.-.	j .---	q --.-	x -..-
d -..	k -.-	r .-.	y -.--
e .	l .-..	s ...	z --..
f ..-.	m --	t -	
g --.	n -.	u ..-	

Se usan todas las combinaciones de entre 1 y 4 puntos y rayas, a excepción de:

..--	.-.-	---.	----

Tradicionalmente, los puntos se transmitían como una nota breve y los guiones como una nota más larga, con pausas entre las diferentes letras. Por eso, algunos teléfonos móviles emiten el sonido ... - ... cuando se recibe un mensaje, ya que este es el código Morse para SMS.

Si los espacios entre las letras se pierden, los mensajes pueden ser ambiguos. Por ejemplo, incluso si sabemos que el mensaje -..-----. se compone de tres letras, aún podría significar: njg, dog, xmg o xon.

Realice un programa que lea un mensaje enviado en código Morse (entre 1 y 10 letras inclusive) determine, y muestre, la cantidad de mensajes con el mismo número de letras que podría representar.

ARREGLOS LINEALES

Debe considerar que los arreglos lineales (también conocidos como vectores) poseen **N** componentes o casillas, donde el valor de **N** debe ser entero y positivo (aunque esto puede cambiar según el lenguaje de programación que se use), y en general el mismo debe ser solicitado al usuario y ser validado adecuadamente para que sea consistente con las declaraciones realizadas en el programa. Los arreglos lineales son homogéneos en su composición, ya que todas las casillas son del mismo tipo de dato.

Dejamos a su criterio de donde serán leídos los valores que componen el vector, pudiendo ser teclado, desde un archivo, o generados de forma aleatoria.

Si bien es cierto que los problemas propuestos en esta sección, pueden ser realizados en un programa, se recomienda que los cálculos sean realizados en rutinas y/o procedimientos en las cuales también se recomiendan diseñar con pase de parámetros.

En la siguiente sección se presentan enunciados que se indica específicamente el uso de arreglos lineales:

116. Dado un arreglo lineal de **N** componentes numéricos, realice un programa que indique el menor valor contenido en el mismo. Considere que el vector no necesariamente se encuentra ordenado.
117. Diseñe un programa que contenga una rutina que dado un vector A de **N** componentes numéricos, genere un resultado **True** si el mismo está **ordenado** y **False** en caso contrario. Considere que el ordenamiento puede ser de mayor a menor o de menor a mayor y que pueden existir elementos repetidos.
118. Diseñe un programa que contenga una rutina que dado un vector A de N componentes numéricos, genere un resultado **True** si el mismo es **simétrico** con respecto a la posición central, en caso contrario el resultado deberá ser **False**.
119. La Media Aritmética (también llamada Promedio o simplemente Media), se obtiene a partir de la suma de todos sus valores dividida entre el número de sumandos Dado un arreglo lineal de **N** componentes numéricos, realice una rutina que calcule la **Media** de sus componentes.
120. Realice un programa que dado un vector de **N** componentes numéricos calcule su **módulo** o **norma**.
121. Dados dos arreglos lineales A y B de **N** componentes numéricos cada uno, realice una rutina que calcule el **producto escalar** de los mismos.
122. Dados dos vectores de N componentes numéricos realice un programa que calcule **el ángulo que existe entre ellos**, muestre el resultado en grados.
123. Realice una rutina que devuelva la primera posición del primer elemento que se **repita** dentro de un arreglo lineal de **N** posiciones.
124. Desarrolle un programa que ordene de **menor a mayor** un arreglo unidimensional A de **N** componentes. El resultado debe ser dado en el mismo vector original y considere como limitante no poder emplear vectores auxiliares. Investigue y realice rutinas para los siguientes enfoques de ordenamiento: Método de la **Burbuja**, Método **Shell**, Método **Quick Sor**, Método de **Inserción Directa**, Método de **Inserción Binaria**, y el Método de **Selección**.

125. Dado un arreglo unidimensional de **N** elementos numéricos y una variable cualquiera Y. Determine si existe algún elemento del arreglo que **sea igual** a Y. De ser así, presentar en qué posición o posiciones se da esta situación.

126. Dado un arreglo unidimensional de **N** componentes numéricos genere una rutina que **ordene** los elementos de las **posiciones pares** de menor a mayor y los elementos de las **posiciones impares** de mayor a menor. El resultado debe ser dado en el mismo vector original y considere como limitante no poder emplear vectores auxiliares.

127. Dado el arreglo lineal A con **N** elementos numéricos enteros, diseñe un programa para construir **otro arreglo** lineal P que contenga todos los elementos de A que sean **números primos**.

128. Dado un conjunto de vectores de N componentes numéricos cada uno, realice un programa que indique si los mismos son **linealmente dependientes**.

129. Dado un arreglo lineal A de **N** componentes numéricos, diseñe una rutina que indique si este vector es lo que llamaremos "***tipo montaña***" y esto ocurre cuando todos los valores van en un riguroso aumento y luego de llegar a un máximo los siguientes componentes van descendiendo con la misma rigurosidad. Considere una variante donde se tendría un vector "***tipo meseta***" si el valor máximo se encuentra repetido uno al lado del otro.

130. Dado un arreglo lineal A de **N** componentes, diseñe un programa donde sea "***eliminado***" un valor seleccionado por el usuario. Debe entenderse por "***eliminar***" que se debe obtener que el arreglo A ahora no tenga el elemento indicado (tantos como se encuentre) y que su dimensionamiento se disminuya en la cantidad pertinente. Por ejemplo para el siguiente vector de dimensión N = 6, se desea eliminar el valor 4:

Vector original	12	4	8	4	25	36
Vector Resultante	12	8	25	36		

131. Dado un vector A de **N** componentes y un vector B de M componentes, diseñe un programa que le permita obtener un tercer vector C que sea la **unión** de los vectores A y B. Como condición debe prever que el resultado **no** debe presentar **valores repetidos**.

132. Dado un arreglo lineal A de **N** componentes y un vector B de **M** componentes, realice un programa que le permita obtener un tercer vector C que posea sólo los **elementos comunes** entre los vectores A y B. Como condición debe velar para que el resultado **no** presente **valores repetidos**.

133. Dado un grupo de números enteros y positivos, ordénelos de mayor a menor según el primer dígito (por ejemplo, para cuatro números: 2950, 3200, 91 y 456 al ser ordenados de esta manera quedarían: 91, 456, 3200 y 2950). Trabaje con valores numéricos y no use funciones reservadas para valores alfanuméricos.

134. Diseñe un programa que dado un valor suministrado por el usuario, **inserte** dicho dato en la posición que le corresponde dentro de un arreglo A de **N** componentes, así que el nuevo valor debe ser ubicado en una posición de tal manera que no rompa el orden.

135. La mediana de un arreglo ordenado de números se puede definir así:
    - Para un número impar de elementos, la *mediana* es el valor que ocupa la casilla central.
    - Para un número par de elementos, es el promedio de los valores que se hallan en las casillas adyacentes a la mitad.

    Ejemplo:

Para un número impar de elementos (N =5)

2	5	6	8	9

Mediana = 6 (elemento que ocupa la casilla 3)
Para un número par de elementos (N =6)

2	5	6	8	9	12

Mediana = 7 (promedio de los valores que ocupan las casillas 3 y 4)

Construya un programa que lea **N** valores numéricos y los almacene en un arreglo unidimensional, ordene el arreglo de forma creciente, calcule la **mediana** de los valores contenidos en el arreglo, y finalmente indique el valor de la *mediana*.

136. Se tiene un conjunto de **N** valores reales (que pueden ser almacenados en un arreglo lineal) y un intervalo [A, B]. Realizar un programa que determine y muestre el **promedio** de aquellos valores mayores que A y el promedio para los valores menores que B.

137. Escribir un programa que a partir de un vector A de **N** elementos, se construya otro arreglo B donde cada elemento B[i] contenga la **posición que ocuparía** el elemento A[i] si estuviera ordenado en forma creciente. Por ejemplo:

14	8	3	2	15	12	7

Vector Original - **A**

6	4	2	1	7	5	3

Vector Resultante - **B**

138. Escribir un programa que contenga una rutina que dado un vector de **N** posiciones ordene de mayor a menor los valores (NO las posiciones) pares y de menor a mayor los valores (nuevamente NO las posiciones) impares. El resultado se debe guardar en el mismo vector, manteniendo las posiciones que tenían los valores pares con valores pares y las impares con las impares.

139. El calendario Gregoriano es el que más conocemos y usualmente empleamos, en este tenemos los meses que contienen una determinada cantidad de días, por otro lado, en una variación del calendario Juliano, este no posee meses y solamente se contabilizan los días en secuencia, esto es que el día 15 de enero es el día 15 en el calendario Juliano, el día 10 de febrero será el día 41 y así sucesivamente. Realice un programa que dado un día y mes del calendario Gregoriano calcule su equivalente en el Juliano, y viceversa (para lo que se recomienda dos rutinas). Recuerde tomar en cuenta el caso de años bisiestos.

140. Se pueden almacenar los coeficientes de un polinomio en un arreglo lineal de N posiciones, en concordancia con un polinomio de orden N:

$$A_N X^N + A_{N-1} X^{N-1} + A_{N-2} X^{N-2} + \ldots + A_1 X + A_0$$

Se sabe que su derivada y su integral son, respectivamente:

D: $$N A_N X^{N-1} + (N-1) A_{N-1} X^{N-2} + (N-2) A_{N-2} X^{N-3} + \ldots + A_1$$

$$\frac{A_N}{N+1} X^{N+1} + \frac{A_{N-1}}{N} X^N + \frac{A_{N-2}}{N-1} X^{N-1} + \ldots + \frac{A_1}{2} X^2 + A_0 X$$

Diseñe un programa que:

- Lea el grado del polinomio **N**, los coeficientes del mismo y tres valores **A**, **B** y **C**.
- Calcule la ***derivada*** del polinomio evaluada en el valor **A**.
- Calcule la ***integral*** del polinomio definida en el intervalo (**B**, **C**).

141. Realice un programa que contenga una rutina en que dado dos números escritos en base dos (también conocida como notación binaria), los sume respetando las siguientes limitaciones:

    - los dígitos de los números deben ser almacenados en vectores,
    - considere números con no más de diez (16) dígitos binarios,
    - tome en cuenta que no necesariamente los dos números poseen la misma cantidad de dígitos,

    Se debe realizar la suma en binario, no se aceptará que se conviertan los números de base para realizar la operación y luego revertir la base del resultado.

142. En estadística descriptiva, se define el rango de un conjunto de datos reales como la diferencia entre el mayor y el menor de los datos.

    Por ejemplo, si los datos son:

    [5.96, 6.74, ***7.43***, 4.99, 7.20, ***0.56***, 2.80], entonces el rango es 7.43 − 0.56 = **6.87**

    Escriba un programa que:

    - pregunte al usuario cuántos datos serán ingresados,
    - pida al usuario ingresar los datos uno por uno, y
    - entregue como resultado el rango de los datos.

ARREGLOS BIDIMENSIONALES

Los arreglos bidimensionales, también conocidos como matrices, son disposiciones de valores homogéneos en su tipo en una matriz, tome por convención que el primer índice corresponde a las filas y el segundo índice corresponde a la columna, así una matriz de orden NxM tendrá N filas y M columnas. En general se trabaja con dos dimensiones, pero la mayoría de los lenguajes permiten trabajar con cuantas dimensiones necesite.

Dejamos a su criterio de donde serán leídos los valores que componen la matriz, pudiendo ser teclado, desde un archivo, o generados de forma aleatoria; con igual consideración para presentar las matrices resultantes, en aquellos problemas cuyo resultado también sea una matriz.

Los siguientes problemas son para generar, modificar u obtener resultados de matrices:

143. Realice un programa que contenga una rutina que **sume** dos (2) matrices de orden NxM. Recuerde que para cualquier elemento ***i,j*** la suma pedida será **C**(i,j) = **A**(i,j) + **B**(i,j)

144. Definamos que una **matriz pináculo** es aquella matriz cuadrada de orden impar que posee el valor más alto en toda la matriz, en la posición central, esto es, la intersección entre la diagonal principal y la diagonal secundaria.

    Conocida la definición anterior realice un programa que indique si una matriz dada es o no *matriz pináculo*, así mismo debe validar el valor de la dimensión de la matriz.

145. Realice un programa que dada una matriz A de orden NxM indique cual es el **mayor valor** que almacena, señalando la posición del mismo; considere la posibilidad de que dicho valor se encuentre repetido, por lo cual deberá indicar todas las posiciones.

146. Genere un programa que posea una rutina que dada una matriz de orden NxM genere su **transpuesta**.

147. Realice un programa que contenga una rutina que **multiplique** dos (2) matrices, se deberá validar los órdenes de las matrices para que sea posible la multiplicación, o en su defecto se pueda ofrecer un mensaje adecuado.

    El resultado de multiplicar dos matrices es otra matriz en la que el elemento que ocupa el lugar C_{ij} se obtiene sumando los productos parciales que se obtienen al multiplicar todos los elementos de la fila "i" de la primera matriz por los elementos de la columna "j" de la segunda matriz.

148. Teniendo una matriz A de orden NxM, desarrolle un programa que indique si tal matriz es **simétrica** respecto a la diagonal principal o no. Como variante diseñe un programa similar, pero en este caso que la simetría sea con respecto a la diagonal secundaria.

149. Se dice que una matriz a es **antisimétrica** si para todo par de índices i y j se cumple que a[i, j] = -a[j, i] (note el signo menos). Diseñe un programa que posea una rutina que dada una matriz A de orden NxN indique si la misma es o no, antisimétrica.

150. Dada una matriz numérica, se denomina "**Elemento Punto de Silla**" aquel valor que es simultáneamente máximo de su fila y mínimo de su columna. Diseñe un programa que determine y muestre todos los puntos de silla de una matriz de orden NxM.

151. La **norma** de una matriz A cuadrada, de orden N, se determina de la siguiente forma:

$$\|A\|_{\infty} = Max(1 < i < n)\sum_{j=1}^{n}\left|a_{ij}\right|$$

En forma operativa se obtiene del siguiente ejemplo de una matriz en donde el usuario ofreció los valores mostrados de una matriz de 3*3:

1	2	-1
0	3	-2
5	-2	1

Tendremos entonces:

|1| + |2| + |-1| = 4 , |0| + |3| + |-2| = 5, |5| + |-2| + |1| = 8

En este caso se toma el máximo {4, 5, 8} que es 8. Diseñe un programa que dada una matriz de orden **N** calcule su norma según el proceso explicado antes

152. Suponga que llega a tener un sistema de N ecuaciones con N incógnitas reducido tal como se muestra (se recomienda estudiar el **método de Gauss**, que consiste en transformar el sistema dado en otro equivalente), en este caso los elementos ubicados a la izquierda de la diagonal secundaria son nulos:

$$\begin{array}{lllllll}
0 & 0 & 0 & 0 & A_{1,N}X_N & = & B_1 \\
0 & 0 & 0 & A_{2,N-1}X_{N-1} + A_{2,N}X_N & & = & B_2 \\
0 & 0 & A_{3,N-2}X_{N-2} + A_{3,N-1}X_{N-1} + A_{3,N}X_N & & & = & B_3 \\
0 & 0 & & & & \bullet & \\
 & & & & & \bullet & \\
0 & A_{N-1,2}X_2 + \ldots + A_{N-1,N-2}X_{N-2} + A_{N-1,N-1}X_{N-1} + A_{N-1,N}X_N & & & & = & B_{N-1} \\
A_{N,1}X_1 + A_{N,2}X_2 + \ldots + A_{N,N-2}X_{N-2} + A_{N,N-1}X_{N-1} + A_{N,N}X_N & & & & & = & B_N
\end{array}$$

Diseñe un programa que:

- Lea y valide adecuadamente un valor **N**.
- Lea la matriz de coeficientes **A** (ya reducida) y el vector de términos independientes **B**. Evite leer los elementos de la matriz **A** ubicados a la izquierda de la diagonal secundaria.
- Si un elemento de la diagonal secundaria es nulo, detenga el proceso.
- Halle y escriba el vector solución **X**.

153. Con los mismos lineamientos y consideraciones del problema anterior, se le presenta un **sistema triangular** de orden N en el estilo del mostrado (es de orden 4, solo como ejemplo):

$$\begin{array}{llllll}
A_{11}X_1 & A_{12}X_2 & A_{13}X_3 & A_{14}X_4 & = & B_1 \\
A_{21}X_1 & A_{22}X_2 & A_{23}X_3 & 0 & = & B_2 \\
A_{31}X_1 & A_{32}X_2 & 0 & 0 & = & B_3 \\
A_{41}X_1 & 0 & 0 & 0 & = & B_4
\end{array}$$

Diseñe un programa que contenga una rutina que resuelva un sistema como el planteado.

154. Diseñe un programa que genere una matriz cuadrada de orden **N** (que debe ser par) y es un dato suministrado por el usuario; según el patrón mostrado para un ejemplo de N = 6.

1	2	3	4	5	6
13	14	15	16	17	18

25	26	27	28	29	30
31	32	33	34	35	36
19	20	21	22	23	24
7	8	9	10	11	12

155. Realice un programa que genere una matriz A de orden NxM en zig-zag horizontal según la secuencia que se muestra en el ejemplo:

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16

El ejemplo mostrado es para una matriz de cuatro (4) filas y cinco (5) columnas, pero recuerde que su propuesta debe ser general.

156. Realizar un programa que contenga procedimientos que realicen y muestren el llenado de una matriz para cada una de las siguientes formas:

A					B					C				
1	2	3	4	5	**1**	2	3	4	5	**25**	23	19	13	5
16	17	18	19	6	10	11	12	13	6	22	18	12	4	9
15	24	**25**	20	7	17	18	19	14	7	17	11	3	8	16
14	23	22	21	8	22	23	20	15	8	10	2	7	15	21
13	12	11	10	9	**25**	24	21	16	9	**1**	6	14	20	24

NOTA: Las matrices anteriores son de ejemplo y el programa debe ser capaz de generar cualquier matriz de orden **N,** donde **N** es leído como dato.

157. Realice un programa que genere y presente una matriz cuadrada A de orden **N** (donde N es dato), de la siguiente manera:

Por ejemplo, para N = 5:

1	2	4	7	11
3	5	8	12	16
6	9	13	17	20
10	14	18	21	23
15	19	22	24	25

158. Diseñe un programa para construir y presentar una matriz cuadrada A de orden **N** (donde N es dato), de la siguiente manera:

Por ejemplo para N = 5:

1	2	3	4	5
2	1	2	3	4
3	2	1	2	3
4	3	2	1	2
5	4	3	2	1

159. Diseñe un programa que genere y presente una matriz cuadrada de orden **N** (que debe ser par) según el patrón mostrado para un ejemplo de N = 6.

1	13	25	36	24	12
2	14	26	35	23	11

3	15	27	34	22	10
4	16	28	33	21	9
5	17	29	32	20	8
6	18	30	31	19	7

160. Diseñe un programa que genere y presente una matriz cuadrada de orden **N** (que debe ser par) según el patrón mostrado para un ejemplo de N = 6.

1	3	6	10	15	21
2	5	9	14	20	26
4	8	13	19	25	30
7	12	18	24	29	33
11	17	23	28	32	35
16	22	27	31	34	36

161. El "**Rectángulo de Tartaglia**" es una forma de representar el Triángulo de Pascal, su estructura es la siguiente:

1	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56
1	5	15	35	70	126
1	6	21	56	126	252
1	7	28	84	210	462
1	8	36	120	330	792
			:		

Elabore un programa que dado un número entero y positivo **N**, genere las primeras N líneas del Rectángulo de Tartaglia.

162. Un **triángulo de Parkside** está compuesto de **N columnas** que contienen sucesiones periódicas de los números del 1 al 9, comenzando la primera sucesión en un valor **X** (denominado valor semilla). Así dados los siguientes valores de N y X se forman los correspondientes triángulos, a continuación, se muestran dos ejemplos:

```
N= 5  X= 3          N= 4  X= 8
3  4  6  9  4       8  9  2  5
      5  7  1  5          1  3  6
         8  2  6             4  7
            3  7                8
               8
```

Escriba un programa que genere triángulos de **Parkside** dados los valores de **N** y **X**.

163. Realice un programa que a partir de una matriz de orden **NxM**, genere dos vectores: el primero de orden N donde cada componente corresponde a la suma de cada fila y otro vector de orden M, formado por la suma de los elementos de cada columna de la matriz.

Por ejemplo, para una matriz de 3x4 se tendría lo siguiente:

1	3	7	2	13

5	6	9	1	21
4	1	7	2	14
1	4	6	5	16
11	14	29	10	

164. Genere un programa que dada una matriz cuadrada (de orden impar), la rote siguiendo el esquema mostrado en el ejemplo para una matriz de orden N= 5 (note que no se modifican las diagonales):

Matriz Original

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Matriz Resultante

1	16	11	6	5
22	7	12	9	2
23	18	13	8	3
24	17	14	19	4
21	10	15	20	25

165. Una forma de generar **cuadrados mágicos** de orden impar se describe a continuación (en este caso se emplea una matriz de orden igual a 3):

- Coloque un 1 en la columna central de la primera fila

	1	

- Muévase en diagonal una casilla hacia arriba y una hacia la derecha, eventualmente este movimiento hará salir de la matriz, en este caso se coloca el número en la última fila en la columna correspondiente.

	1	
		2

- Continuando el movimiento (una columna hacia la derecha y una fila hacia arriba), abandonará eventualmente la matriz por su lado derecho, así que coloca el valor en la primea columna en la fila correspondiente.

	1	
3		
		2

- En algún momento se tratará de llegar a una casilla ya escrita (esto ocurre con todos los múltiplos del orden de la matriz). En este caso se desciende una fila y se mantiene en la misma columna desde la posición original.

	1	
3		
4		2

- Como una variante del caso anterior se tiene que tener especial atención al caso que se presentará siempre en la casilla de la primera fila y última columna.

8	1	6
3	5	7
4	9	2

Diseñe un programa que solicite y valide adecuadamente el orden del cuadrado mágico a ser generado, y desarrolle el método antes explicado presentando el resultado.

166. Desarrolle un programa que contenga una rutina que genere el resultado True (Verdadero) si una matriz cuadrada, pasada como parámetro, **forma Cuadrado Mágico**, en caso contrario el resultado de la rutina será False (Falso). Entendiéndose como cuadrado mágico aquella matriz que la suma de los elementos de cada fila da el mismo valor, que a su vez es igual la suma de los elementos de cada columna, que también es igual a la suma de los elementos de la diagonal principal y los de la secundaria. En el programa se leerá la matriz, se invocará la rutina solicitada y se dará el mensaje pertinente.

167. En una matriz de 8 x 8 se puede representar un **tablero de ajedrez**, por ejemplo, los valores en cero significan casillas vacías, casillas con valores positivos alojan piezas de color blanco y casillas con valores negativos son asiento de las piezas negras, esto. Diseñe un programa que para una **torre** de color blanco ubicada en un valor de **Fila** y **Columna** dados como datos, debe indicar cuántas piezas contrarias puede tomar. Es pertinente considerar que las posiciones de todas las piezas en el tablero deben ser datos de este programa.

168. Piense en el mismo problema anterior, sólo que ahora la pieza a ser evaluada será un **alfil**.

169. Bajo los lineamientos de los problemas anteriores, conocida la posición que ocupa un **caballo** en un tablero de ajedrez por las coordenadas del cuadro donde se encuentra, determinar las posiciones a las cuales puede moverse.

170. Realice una **rutina** que dado un valor **N**, genere una matriz cuadrada de valores enteros y que su relleno siga la ley de mostrada en el ejemplo anexo, el valor de **N** no debe ser menor a 7, debe ser impar y su valor máximo será de 99 (todas estas condiciones debe evaluarlas y pasar los parámetros y sus valores pertinentes para un buen funcionamiento de la rutina.

Por ejemplo, para un valor de N = 9, se obtiene la siguiente matriz:

1	0	0	0	0	0	0	0	9
10	2	0	0	0	0	0	8	18
19	11	3	0	0	0	7	17	27
28	20	12	4	0	6	16	26	36
37	29	21	13	5	15	25	35	45
46	38	30	22	0	24	34	44	54
55	47	39	0	0	0	43	53	63
64	56	0	0	0	0	0	62	72
73	0	0	0	0	0	0	0	81

ARCHIVOS TIPO TEXTO

El uso de archivos tipo texto (muchas veces llamados archivos planos por una mala traducción del inglés) es acudir a una herramienta que permite guardar de forma permanente datos o resultados en un dispositivo de almacenaje diferente a la memoria principal (disco duro, pendrive, etc).

Se debe separar problemas asociados al manejo de archivos en dos aspectos básicos, uno es el resolver el problema propiamente dicho, y el otro es la gestión de lectura y escritura de los valores en el archivo. En el primer caso

este tipo de almacenaje es muy empleado en el trabajo con arreglos lineales (vectores) o con arreglos bidimensionales (matrices), así que puede replantear los problemas propuestos en esas secciones leyendo o escribiendo los resultados en archivos.

171. Para el cálculo de la Media, la Varianza y la Desviación Estándar de un conjunto de **N** valores, se emplean las fórmulas mostradas en la columna de la izquierda:

$$X_m = \frac{\sum_{i=1}^{N} X_i}{N}$$

$$VAR = \frac{\sum_{i=1}^{N} \left(X_i - X_m \right)^2}{N}$$

$$DES = \sqrt{VAR}$$

DATOS.DAT

N :(# de datos)
X1
X2
X3
.
.
X_N

Realizar un programa que lea un archivo tipo texto 'Datos.Dat', el cual posee la estructura mostrada en la columna de la derecha (note que el número de datos es un valor escrito en la primera línea), y que calcule y presente los valores de la Media, la Varianza y la Desviación Estándar de ese conjunto de datos.

172. Realice el problema anterior, pero sin conocer a priori la cantidad de valores que hay dentro del archivo.

173. Diseñar un programa que escriba en un archivo tipo texto todos los pares de números (cada par en una línea), cuyos valores no excedan de un valor **N** (suministrado por el usuario), tales que su **cociente** sea múltiplo de tres o siete, y su **producto** termine en dos. Por ejemplo: el par 24 y 8 cumplen con lo pedido ya que 24/8 = 3 y 24*8=192

174. En un archivo tipo texto tiene las coordenadas (**x, y, z**) de un número no determinado de puntos en el espacio. Realice un programa que solicite al usuario un valor real que corresponderá al radio de una esfera cuyo centro se encuentra en el origen y que calcule los porcentajes de puntos que se encuentran dentro, sobre o fuera de dicha esfera. Tales valores deben ser anexados al final del archivo dado.

175. Realice el problema anterior, pero como variante considere que la esfera tiene su centro en unas coordenadas también suministradas por el usuario.

176. Se desea enviar un mensaje en un archivo tipo texto de la siguiente forma: para cada palabra **W** se selecciona un número **N** y se replica **W N** veces horizontalmente. Luego la cadena de caracteres se repite en la siguiente línea, pero esta vez rotando los caracteres una vez a la izquierda. Este proceso se repite hasta la línea antes que la palabra **W** aparezca escrita en la primera columna del patrón rectangular producido por este procedimiento.

Por ejemplo, cuando se escoge la palabra **W** = *Hola* y el número **N** = *3*, debemos producir el siguiente patrón:

HolaHolaHola
olaHolaHolaH
laHolaHolaHo

aHolaHolaHol…

Como entrada se tendrá un archivo tipo texto que contiene información de una cantidad indeterminada de palabras en líneas separadas antecedidas por un número entero menor a 10. Como salida debe generarse un archivo texto codificado de la forma anteriormente descrita.

Ejemplo de entrada:	Ejemplo de salida:
2 Delphi	DelphiDelphi
4 Quiz	elphiDelphiD
	lphiDelphiDe
	phiDelphiDel
	hiDelphiDelp
	iDelphiDelph
	QuizQuizQuizQuiz
	uizQuizQuizQuizQ
	izQuizQuizQuizQu
	zQuizQuizQuizQui

177. Pretende llevar en un archivo tipo texto una **agenda telefónica**, en ella guardará el primer nombre, el primer apellido y el número de teléfono; la agenda está organizada de forma alfabética. Realice un programa que solicite los datos de un nuevo contacto y guarde los mismos en el archivo, pero manteniendo el mismo ordenado. Decida la estructura del archivo.

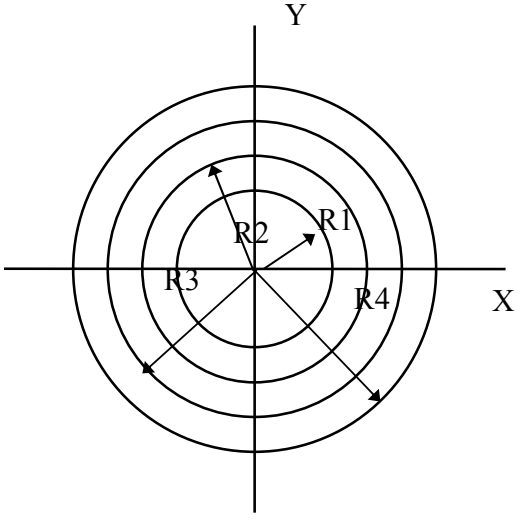
178. Realizar un programa basado en una rutina que **construya un histograma** que represente la longitud de las palabras que se encuentran en un archivo tipo texto. El programa debe tomar cada una de las palabras y por cada letra debe imprimir en otro archivo tipo texto un asterisco, simulando una barra del histograma. **Entrada**: La primera línea del archivo de entrada contiene el número de barras que debe tener el histograma y las siguientes líneas contienen las palabras que deben representarse en el mismo. **Salida**: La salida deseada es un archivo texto con el histograma antes mencionado, el cual debe tener barras representadas por asteriscos correspondientes al número de letras que tiene cada palabra del archivo de entrada.

179. Existe en disco un archivo de datos 'DATOS.DAT' donde cada línea del archivo obedece a la siguiente estructura:

Nombre de una persona, Cédula de Identidad, Estatura

Escriba un programa que reciba por teclado un número de Cédula de Identidad, **busque** en el archivo este valor y muestre en pantalla todos los datos correspondientes a esa persona. Si el número de Cédula no se encuentra en el archivo, el programa debe señalarlo con un mensaje.

180. Dos personas juegan a los dardos con el blanco que se muestra en la siguiente figura:



El puntaje es como sigue:

Zona I: 100 puntos
Zona II: 25 puntos
Zona III: 5 puntos
Zona IV: 1 punto

Diseñe un programa que, utilizando una sola instrucción de decisión, indique la persona ganadora. La información de los lanzamientos realizados por los jugadores se encuentra en un archivo, estos som los diferentes radios y las coordenadas *x, y* del lanzamiento de cada dardo para cada jugador y todos estos datos serán leídos desde un archivo tipo texto (Ud. Decide la estructura del mismo).

181. Se necesita realizar un programa que compare distintos valores que pueden tener diferentes bases (en nuestro caso sólo serán base binaria o base decimal). Para ello se da un archivo texto con un número indeterminado de líneas, donde en cada línea habrán cuatro números enteros (a,b,c,d), donde a y c representan los dos números a comparar y b y d serán sus respectivas bases. Se desea producir un el archivo de salida en el cual por cada línea del archivo de entrada se escriba el carácter '>', '<' o '=', dependiendo si "a" es mayor, menor o igual a "c" (tomando en cuenta sus bases).
Ejemplo de archivo de entrada:

10	10	12	10
1011	2	9	10
12	10	1100	2
101	2	101	2
1001	10	1001	2

182. En las elecciones de presidentes de un país de ***N*** estados, se presentaron ***M*** candidatos, se envían como datos en archivos tipo texto: el número del estado y el número de votos obtenidos por cada candidato en ese estado, donde el candidato está identificado por un número entero positivo.

Diseñe un programa que, a partir de la información suministrada, realicen lo siguiente:

a) Calcular los votos obtenidos por cada candidato en el país.
b) Determinar el candidato ganador.
c) Para el candidato ganador determinar el estado donde obtuvo mayor porcentaje de aceptación (votos obtenidos por ese candidato en ese estado, entre votos totales de ese estado).
d) Determine el candidato ganador para cada estado.
e) Escriba los resultados obtenidos en otro archivo tipo texto

PROBLEMAS MÁS INTERESANTES

A continuación, tiene una serie de enunciados que consideramos de mayor interés a los previamente presentados, ya que implica que deduzca y/o decida cuáles son las estructuras de las variables (o combinación de las mismas) que son más convenientes para una adecuada solución.

En algunos casos puede ser simple la solución, pero considere que al incrementarse el número de datos, la solución puede escalar considerablemente el tiempo de solución o el consumo de memoria, así que intente optimizar el uso de los recursos.

183. Una pareja de novios se da cita cada día en la terraza de un café al término de su jornada laboral. Ambos llegan entre las 5:00 p.m. y las 5:45 p.m., de manera **equiprobable e independiente**. Permanecen allí durante **T** minutos. Diseñe un programa que con los criterios anteriores permita conocer la **probabilidad** de que la pareja se **encuentre** en dicho café.

184. A un grupo de números naturales se les conoce como **Sociables**, cumplen lo mismo que los números amigos, pero en vez de ir en parejas van en grupos más grandes. Así, la suma de los divisores del primer número da el segundo, la suma de los del segundo da el tercero, y así sucesivamente. La suma de los divisores del último da el primer número de la lista. Por ejemplo: los números 12496, 14288, 15472, 14536 y 14264 son números sociables.

185. El número 585_{10} (base decimal) es igual a 1001001001_2 (base binaria). Nótese que dicho número es **palindrómico** en ambas bases.

    Realice un programa que calcule la **suma** de todos los números menores a un millón que son simultáneamente palindrómicos en base decimal y base binaria (no incluya los ceros a la izquierda de los números).

186. Realice un programa que determine **cuantos billetes y monedas** de cada denominación se deben emplear para cambiarle el cheque a un cliente, de tal manera que el número de billetes y monedas de circulación legal sea el mínimo posible. No tome en cuenta los centavos. La cantidad debe ser verificada no pudiendo ser negativa, poseer valores decimales o exceder el monto de 100.000.000.000, en cualquiera de los casos se debe dar un mensaje adecuado al usuario. El resultado debe ser dado en forma que se muestre la denominación del billete o moneda y la cantidad requerida del mismo, aquellos billetes o monedas que no sean necesarios no deben ser mostrados.

187. **Los números felices** se definen de la siguiente forma: Empezando con un número (dado como dato) entero positivo **N,** se reemplaza el número por la suma de los cuadrados de sus dígitos, y se repite el proceso hasta que el número es igual a 1 o hasta que se entra en un ciclo infinito que no incluye el 1. Los números que al finalizar el proceso terminan con 1, son conocidos como números felices. Por ejemplo, el 7 es un número feliz porque el proceso finaliza en uno, pero el 4 no es feliz por entra en un ciclo infinito que no contiene al 1.

    Ejemplo para el 7 (*número feliz*):

    $\mathbf{7}^2 = 49$

    $4^2 + 9^2 = 97$

    $9^2 + 7^2 = 130$

    $1^2 + 3^2 + 0^2 = 10$

    $1^2 + 0^2 = \mathbf{1}$

    Ejemplo para el 4 (*numero infeliz*):

    $\mathbf{4}^{2 =} 16$

    $1^2 + 6^2 = 37$

    $3^2 + 7^2 = 58$

    $5^2 + 8^2 = 89$

    $8^2 + 9^2 = 145$

    $1^2 + 4^2 + 5^2 = 42$

    $4^2 + 2^2 = 20$

$2^2 + 0^2 = \mathbf{4}$ ...

188. Suponga que está elaborando un sistema que maneje todos los aspectos relacionados con una nómina en el cual gestiona recibos de pagos de diferentes fuentes (nómina, vacaciones, etc). Como soporte a lo anterior genere un programa que, **dada una cifra**, ingresada por el usuario en números, **sea representada en palabras**. Como limitante el programa debe ser realizado para cifras de no más de nueve (9) dígitos y trabajaremos sin centavos (obviamente estos elementos deben ser validados adecuadamente).

189. La **criba de Eratóstenes** es un algoritmo que permite hallar todos los números primos menores que un número natural dado **N**. Se forma una tabla con todos los números naturales comprendidos entre 2 y n, y se van tachando los números que no son primos de la siguiente manera: Comenzando por el 2, se tachan todos sus múltiplos; comenzando de nuevo, cuando se encuentra un número entero que no ha sido tachado, ese número es declarado primo, y se procede a tachar todos sus múltiplos, así sucesivamente. El proceso termina cuando el cuadrado del mayor número confirmado como primo es mayor que n. Diseñe un programa que dado un número entero y positivo realice el procedimiento descrito y muestre los valores encontrados.

190. Escriba un programa que contenga una **función recursiva** que genere los **números binarios** de **N** bits que no tengan ceros consecutivos. Por ejemplo, para N = 3, los números son: 110, 110, 011 y 101.

191. La **compresión RLE** o *Run-Length Encoding* es una forma muy simple de compresión de datos en la que secuencias de datos con el mismo valor consecutivas son almacenadas como un único valor más su recuento. Esto es más útil en datos que contienen muchas de estas "secuencias"; por ejemplo, gráficos sencillos con áreas de color plano, como iconos y logotipos.
Por ejemplo, considere una pantalla que contiene texto en negro sobre un fondo blanco. Habría muchas secuencias de este tipo con píxeles blancos en los márgenes vacíos, y otras secuencias de píxeles negros en la zona del texto. Supongamos una única línea (o scanline), con 0 representando las zonas en negro y 1 representa las de blanco sería de la siguiente forma: 11111111111101111111111110001111111111111111 (secuencia descomprimida)
Si aplicamos la codificación run-length a ésta línea, obtendríamos lo siguiente: 1(12)0(1)1(12)0(3)1(15).
Interpretado esto como 12 blancos, 1 negro, 12 blancos, 3 negros y 15 blancos.
Esta codificación traducida a binario, cuyo principio es el mismo, se utiliza para el almacenamiento de imágenes. Una forma de hacerlo es utilizando 5 bits, donde: 1 bit (0 o 1) es para el color y 4 bits para especificar el número de caracteres consecutivos, por lo que el ejemplo anterior quedaría representado de la siguiente forma: 1(1100)0(0001)1(1100)0(0011)1(1111) o lo que viene a ser lo mismo 1110000001111000001111111 (secuencia comprimida), con lo que la secuencia inicial de 43 bits queda reducida a 25 bits.
Realice un programa que dada una cadena de caracteres con una secuencia binaria de datos comprimida, produzca la secuencia de datos descomprimida también en cadena de caracteres.

192. Realice un programa que indique los tres últimos dígitos antes de la coma decimal para la expresión: (3 + raíz(5)) elevado a **N**. Por ejemplo, para n = 5,

se tiene que (3 + √5)5 = 3935.73982 y la respuesta debe ser 935. En otro caso para N = 2, se tiene que (3 + √5)2 = 27.4164079 y la respuesta es 027.

193. Realice un programa que contenga una función que determine el **mayor producto de cuatro números adyacentes** (horizontal, vertical o diagonalmente) en una matriz cuadrada de orden **NxN** (N>4). Por ejemplo, en la siguiente matriz de 6x6 el mayor producto es 1788696, dado por la multiplicación de los números marcados en rojo:

02	44	75	33	53	78
10	26	38	40	67	59
20	95	64	94	39	63
26	97	17	78	78	96
44	20	45	35	14	00
67	15	94	03	80	04

194. Un número natural se le denomina **Semiperfecto** si cumple a ser igual a la suma de algunos de sus divisores propios. Por ejemplo, 18 es semiperfecto ya que sus divisores son 1, 2, 3, 6, 9 y se cumple que 3+6+9=18. Diseñe un programa que indique si un valor dado como dato cumple con la definición ofrecida.

195. Realice un programa que calcule la suma de todos los dígitos de los números del 1 al **N** (siendo N un valor dado por el usuario). Como recomendación, no se deje engañar por la aparente simpleza de lo solicitado, ya que una propuesta ineficiente puede hacer que la solución planteada tarde mucho tiempo en dar el resultado a medida que el valor de N sea mayor.

196. **2050** es el número más pequeño que puede ser dividido por todos los números entre 1 y 10. Realice programa que contenga una función que devuelva el número más pequeño que es divido por todos los números entre 1 y un valor N, ingresado como dato por el usuario.

197. Genere un programa que dado un número entre 0,0001 y 0,9999 (y de no más de 4 cifras decimales), obtenga y muestre la correspondiente **fracción irreducible**.

    Por ejemplo, el número 0,25 se puede obtener a partir de 25/100, o de 2/8, o de 1/4, entre otros. La fracción irreducible es 1/4, que está formada por un numerador y un denominador que son primos entre sí.

198. Se tiene un reloj digital que muestra la hora con LED (dos LED para horas, dos para minutos y dos para los segundos) de 7segmentos para cada LED. Diseñe un programa que le indique ¿cuántos segmentos se han encendido DESPUÉS DE X segundos, contando desde la posición 00:00:00?

    Se debe considerar que en cada segundo, todos los LED se apagan y luego se encienden los correspondientes al instante actual.

199. Se está estudiando un enjambre de **N** luciérnagas. Cada luciérnaga se mueve en línea recta a velocidad constante. Asuma que usted se encuentra en el centro del universo, en la posición (0, 0, 0). Todas las luciérnagas tienen la misma masa, y se desea saber a qué distancia mínima desde su ubicación (el origen de coordenadas) se llega a encontrar el centro del enjambre.

Se conoce la posición y la velocidad de cada luciérnaga en t = 0, pero sólo se está interesado en instantes t >= 0. Las luciérnagas tienen velocidad constante, y pueden moverse libremente a través de todo el espacio. Siendo M(t) la ubicación del centro de masa de las **N** luciérnagas en el momento t, d(t) la distancia entre su posición y M(t) en el instante t. Debe encontrar el valor mínimo de d(t), **Dmin**

La entrada de datos tendrá estas características: la primera línea de entrada contiene un único entero T, el número de casos de prueba. Cada caso de prueba se inicia con una línea que contiene un entero N, la cantidad de luciérnagas, seguido por N líneas de la forma:

x y z vx vy vz

Cada una de estas líneas se describe una luciérnaga: (x, y, z) es su posición inicial en el instante t = 0, y (vx, vy, vz) es su velocidad.

La salida deseada para cada caso de prueba será

Case #X: dmin tmín

donde X es el número de caso de prueba, a partir de 1. Cualquier respuesta con un error absoluto o relativo de un máximo de 10-5, será aceptable.

Límites a ser considerados:

- Todos los números en la entrada serán enteros. 1<= t <= 100. Los valores de x, y, z, vx, vy y vz será entre -5000 y 5000, ambos inclusive.
- Conjunto de datos pequeño: 3 <= n <= 10
- Conjunto de datos grande: 3 <= n <= 500

200. Cuando un rey abdica, su primogénito hereda el trono y debe recibir, en su coronación, un número que lo identificará para la posteridad. La numeración se considera importante porque, de otro modo, sería difícil diferenciar a reyes con el mismo nombre de una misma dinastía al compartir también apellido. Como consecuencia de esta práctica, es que ante la abdicación de un rey, toca revisar los libros de historia para averiguar su número. Lo que se le pide es la elaboración de un programa que automatice este proceso.

¿Qué se tiene de entrada? El programa recibirá, preferiblemente en un archivo tipo texto, múltiples casos de prueba. Cada uno consta de una primera línea con un número indicando la cantidad de reyes de una determinada dinastía. A continuación, vendra en otra línea, los nombres de todos sus reyes separados por espacio. Después aparecerán dos líneas más, una con la cantidad de sucesores futuros que hay que numerar (al menos uno), y otra con sus nombres separados por espacio. Después aparecerán dos líneas más, una con la cantidad de sucesores futuros que hay que numerar (al menos uno), y otra con sus nombres separados por espacio.

Todos los nombres estarán compuestos de una única palabra de no más de 20 letras del alfabeto inglés, y serán sensibles a mayúsculas. Además, se garantiza que en cada caso de prueba no habrá más de 20 nombres de reyes diferentes.

Salida que se desea es: para cada sucesor de cada caso de prueba se indicará, una línea independiente, el número que le corresponderá. Aunque normalmente se utilizan números romanos, por simplicidad se indicará el número en la notación arábiga tradicional. Después de cada caso de prueba se escribirá una línea en blanco.

201. Se da en un archivo tipo texto un grupo de valores en un número indeterminado de líneas, en cada línea habrá dos valores enteros A y B

(cumpliéndose que: 1 ≤ A ≤ B ≤ 108). Se requiere que realice un programa que produzca otro archivo tipo texto en el cual por cada línea del archivo de entrada escriba una línea en el archivo de salida con diez valores enteros; cada uno de estos valores será el contaje del número de veces que cada dígito (0-9) aparece en todos los números enteros existentes entre A y B (ambos inclusive). Por ejemplo, si los valores en el archivo inicial son 10 y 12, todos los números involucrados serán 10, 11 y 12 y se tendrá que el dígito 0 aparece 1 vez, el 1 aparece 4 veces, el 2 una vez y el resto de los dígitos no aparecen. Por lo tanto, la línea en el archivo de salida será 1 4 1 0 0 0 0 0 0 0. El conteo de cada dígito se debe escribir en orden creciente del 0 al 9.

Ejemplo de archivo de entrada:

10	12
1	9
12	321
5987	6123
12345678	12345679

Ejemplo de archivo de salida:

1	4	1	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1
61	169	163	83	61	61	61	61	61	61
134	58	28	24	23	36	147	24	27	47
0	2	2	2	2	2	2	2	1	1

202. Un **tautograma** es un poema, verso u oración formado por palabras que empiezan por la misma letra. Por ejemplo, las siguientes oraciones son tautogramas:

- Cristina come ciruelas con Carmen Castillejo cuando cantan canciones cubanas
- Pedro Pablo, pobre pintor, pinta preciosos paisajes por poca plata

Dado un archivo texto con un número indeterminado de oraciones, cada una en una línea, realizar un programa que produzca un archivo texto donde cada línea tenga un "Si" o un "No" dependiendo si en el archivo original la oración que se encontraba en esa línea era o no un tautograma.

Ejemplo de archivo de entrada:

```
Antes alegre andaba; agora apenas
Para poder presumir practica previamente
Procurad pensar poco, pero profundamente
 Esto NO es un tautograma
Bebe Benito brandy. Bebe buen borracho
```

Ejemplo de archivo de salida:

```
Si
Si
Si
No
Si
```

203. Diseñe un programa que dado un archivo tipo texto con un número indeterminado de palabras escritas una debajo de la otra, tenga como una salida una matriz NxN que sea una **sopa de letras** que contenga las palabras que se encuentran en el archivo texto. Las palabras deben aparecer aleatoriamente y con la misma probabilidad de ocurrencia en cualquier posición de la matriz y con alguna de las siguientes direcciones: horizontal (izquierda-derecha) o vertical (arriba-abajo). Asuma que todas las palabras pueden ser colocadas en la matriz y recuerde que las palabras se pueden cruzar. El dato de entrada será el tamaño de la matriz (N) y la petición de ubicación del archivo en el sistema.

204. El problema de **los filósofos cenando** es un problema clásico de las ciencias de la computación propuesto por Edsger Dijkstra para representar el problema de la sincronización de procesos en un sistema operativo. Cabe aclarar que la interpretación está basada en pensadores chinos, quienes comían con dos palillos, donde es más lógico que se necesite el del comensal que se siente al lado para poder comer.

    El enunciado del problema es el siguiente: cinco filósofos se sientan alrededor de una mesa y pasan su vida cenando y pensando. Cada filósofo tiene un plato de fideos y un tenedor a la izquierda de su plato. Para comer los fideos son necesarios dos tenedores y cada filósofo sólo puede tomar los que están a su izquierda y derecha. Si cualquier filósofo toma un tenedor y el otro está ocupado, se quedará esperando, con el tenedor en la mano, hasta que pueda tomar el otro tenedor, para luego empezar a comer.

    Si dos filósofos adyacentes intentan tomar el mismo tenedor a una vez, se produce una condición de carrera: ambos compiten por tomar el mismo tenedor, y uno de ellos se queda sin comer.

    Si todos los filósofos toman el tenedor que está a su derecha al mismo tiempo, entonces todos se quedarán esperando eternamente, porque alguien debe liberar el tenedor que les falta. Nadie lo hará porque todos se encuentran en la misma situación (esperando que alguno deje sus tenedores). Entonces los filósofos se morirán de hambre. Este bloqueo mutuo se denomina interbloqueo o deadlock.

    Se pide que investigue y programe alguna de las posibles soluciones que permita que los filósofos nunca se mueran de hambre.

205. Una pequeña editorial pretende poner en marcha una campaña de promoción de la última novela de uno de sus escritores y para ello va a lanzar al mercado dos formatos de presentación de la misma, libro de bolsillo y libro de tapa dura. En el departamento de impresión disponen de 140 horas para su tarea sobre el proyecto y en el departamento de encuadernación de 250. Los ingresos obtenidos por cada libro de bolsillo son de 10 unidades monetarias y para su elaboración se requiere 1 hora en el departamento de impresión y 2 horas en el de encuadernación y los ingresos obtenidos por cada libro de tapa dura, son de 17 unidades monetarias, siendo 2 las horas necesarias para su elaboración en el departamento de impresión y 3 las necesarias en el de encuadernación.

    Realice un programa que determine ¿cuantos libros de cada uno de los formatos ha de publicar la editorial para obtener beneficio máximo? y ¿a cuánto ascienden los ingresos correspondientes?

206. El Juego de la Vida es un autómata celular diseñado por el matemático británico John Horton Conway en 1970. Es un juego de cero jugadores, lo que quiere decir que su evolución está determinada por el estado inicial y no necesita ninguna entrada de datos posterior. El "tablero de juego" es una malla formada por cuadrados ("células") que se extiende por el infinito en todas las direcciones. Cada célula tiene 8 células vecinas, que son las que están próximas a ella, incluyendo las diagonales. Las células tienen dos estados: "vivas" o "muertas" (o "encendidas" y "apagadas"). El estado de la malla evoluciona a lo largo de unidades de tiempo discretas (se podría decir que por turnos). El estado de todas las células se tiene en cuenta para calcular el estado de las mismas al turno siguiente. Todas las células se actualizan simultáneamente.

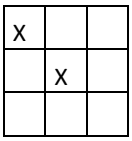
Las transiciones entre unidades de tiempo dependen del número de células vecinas vivas:

- Una célula presente en la generación *t* morirá (desaparecerá) en la generación *t+1* si:
  - Se encontraba rodeada por menos de 2 células (muerte por soledad)
  - Se encontraba rodeada por más de 3 células (muerte por superpoblación)

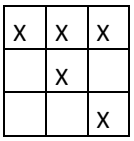
- Una célula nacerá (aparecerá) en la generación *t+1* en una celda vacía si esa celda se encontraba rodeada por exactamente tres células vivas en la generación anterior

- En cualquier otro caso, una celda no variará su estado al pasar de la generación *t* a la *t+1*

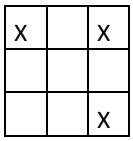
Ejemplos:



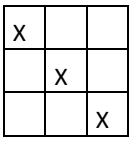
La célula central desaparecerá en la siguiente generación por soledad (1 vecino)



La célula central morirá en la siguiente generación por superpoblación (4 vecinos)

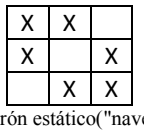


Una nueva célula nacerá en la siguiente generación en la casilla central (3 vecinos)

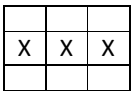


La celda central seguirá conteniendo una célula viva en la siguiente generación (2 vecinos)

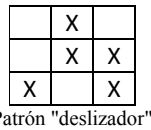
Algunas configuraciones iniciales interesantes son:



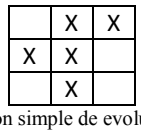
Patrón estático("nave")



Patrón oscilante ("faro")



Patrón "deslizador"



Patrón simple de evolución compleja (R-pentominó)

Es importante recalcar que la supervivencia, nacimiento o muerte de las células deberá determinarse considerando sus vecinos en esa generación. Es decir, no deberán realizarse cambios sobre el tablero hasta que no se haya determinado el destino de todas las celdas en la siguiente generación.

Se pide implementar el Juego de la Vida, propuesto por el matemático americano John Conway.
El juego utilizara un tablero que, para esta implementación, tendrá una dimensión de NxN (con N definido por el usuario). Cada celda de este tablero podrá estar ocupada por una célula viva (con un carácter o color que la represente) o podrá estar vacía para representar una célula muerta. En el juego se parte de una configuración inicial de células (suministrada por el usuario o generada al azar) y se deja evolucionar esta población de acuerdo a las reglas definidas por Conway.

El programa debe mostrar una simulación de la evolución de la población inicial durante T generaciones (parámetro suministrado por el usuario). Tras mostrar por pantalla la población obtenida, el programa permitirá:

a) Terminar el juego.
b) Guardar la configuración obtenida en un archivo texto.
c) Continuar calculando nuevas generaciones.

El programa permitirá definir la configuración inicial por cualquiera de estos procedimientos:

a) Leyendo datos suministrados por el teclado o ratón.
b) Generando aleatoriamente la configuración inicial. En este modo, el usuario fijará porcentaje de celdas ocupadas por células vivas, y el programa distribuirá estas células de forma aleatoria sobre el tablero.
c) Leyendo datos de un archivo texto.

El programa permitirá configurar el comportamiento del juego con células que se encuentren en los bordes del tablero. En particular, será posible seleccionar cualquiera de estas posibilidades:

a) Tablero plano: una celda en el borde del tablero evolucionará como si todos sus vecinos no accesibles fueran celdas vacías.
b) Tablero cilíndrico: en este modo, se tomarán la primera y última columnas del tablero como contiguas.
c) Tablero toroidal: en este modo, se considerarán como contiguas tanto la primera y última columnas del tablero como la primera y última fila.

b. Diseñe una función para hallar una buena aproximación de la longitud de curva de la función y = EXP(-X*X) en un intervalo [a,b] definido por el usuario.

d. Dada una matriz cuadrada de orden impar ordenada como se muestra en el ejemplo (de dimensión 5),

21	22	23	24	**25**
20	**7**	8	**9**	10
19	6	**1**	2	11
18	**5**	4	**3**	12
17	16	15	14	**13**

Realizar un **subprograma** que tenga como parámetro el orden de la matriz y como salida la suma de los valores que están en las diagonales principales de la matriz (para el caso del ejemplo de dimensión 5 es 101)

f. Los ***números cadena*** son creados al sumar continuamente el cuadrado de cada dígito de un número para formar otro. Por ejemplo, si se tiene el número 44, se procede a elevar cada uno de sus dígitos al cuadrado y la suma de ellos resulta en 32 ($4^2 + 4^2 = 16 + 16 = 32$). Luego se hace lo mismo con el 32 ($3^2 + 2^2 = 13$) y así sucesivamente. Por ejemplo:

44 →32 →13 →10 →**1** →**1**
85 →**89** →145 →42 →20 →4 →16 →37 →58 →**89**

Una vez que la cadena llegue a 1 u 89, la cadena queda en un ciclo infinito como los mostrados en los ejemplos anteriores. Es interesante notar que al comenzar con CUALQUIER número entero positivo, eventualmente llegará a 1 u 89.

Se pide que diseñe un programa que permita conocer cuál es el porcentaje de números enteros menor a 10 millones cuya cadena llega al número 89.

g. Diseñe un subprograma que determine el número de veces que aparece una palabra ‘A’ ingresada por un usuario en una cadena de caracteres ‘B’ también ingresada por el usuario.

h. La compresión RLE o Run-length encoding es una forma muy simple de compresión de datos en la que secuencias de datos con el mismo valor consecutivas son almacenadas como un único valor más su recuento. Esto es más útil en datos que contienen muchas de estas "secuencias"; por ejemplo, gráficos sencillos con áreas de color plano, como iconos y logotipos.
Por ejemplo, considere una pantalla que contiene texto en negro sobre un fondo blanco. Habría muchas secuencias de este tipo con píxeles blancos en los márgenes vacíos, y otras secuencias de píxeles negros en la zona del texto. Supongamos una única línea (o scanline), con 0 representando las zonas en negro y 1 las de blanco:
111111111111011111111111100011111111111111 (secuencia descomprimida)
Si aplicamos la codificación run-length a ésta línea, obtendríamos lo siguiente:
1(12)0(1)1(12)0(3)1(15)
Interpretado esto como 12 blancos, 1 negro, 12 blacos, 3 negros y 15 blancos.
Esta codificación traducida a binario, cuyo principio es el mismo, se utiliza para el almacenamiento de imágenes. Una forma de hacerlo es utilizando 5 bits: 1 bit (0 o 1) para el color y 4 bits para especificar el número de caracteres consecutivos, por lo que el ejemplo anterior quedaría representado de la siguiente forma:
1(1100)0(0001)1(1100)0(0011)1(1111) o lo que es lo mismo 11100000011110000011111111 (secuencia comprimida), con lo que la secuencia inicial de 43 bits queda reducida a 25 bits.
Realice una función que dado una cadena de caracteres con una secuencia binaria de datos comprimida, produzca la secuencia de datos descomprimida también en cadena de caracteres.

i – Se da en un archivo tipo texto un grupo de valores en un número indeterminado de líneas, en cada línea habrá dos valores enteros A y B ($1 \leq A \leq B \leq 10^8$). Se requiere que realice un subprograma que produzca otro archivo tipo texto en el cual por cada línea del archivo de entrada escriba una línea en el archivo de salida con diez valores enteros; cada uno de estos valores será el contaje del número de veces que cada dígito (0-9) aparece en todos los números enteros existentes entre A y B (ambos inclusive). Por ejemplo, si los valores en el archivo inicial son 10 y 12, todos los números involucrados serán 10,11y12 y se tendrá que el dígito 0 aparece 1 vez, el 1 aparece 4 veces, el 2 una vez y el resto de los dígitos no aparecen. Por lo tanto la línea en el archivo de salida será 1 4 1 0 0 0 0 0 0 0. El conteo de cada dígito se debe escribir en orden creciente del 0 al 9.

Ejemplo de archivo de entrada:

```
10 12
1 9
12 321
5987 6123
12345678 12345679
```

Ejemplo de archivo de salida:

```
1 4 1 0 0 0 0 0 0 0
```

```
0 1 1 1 1 1 1 1 1
61 169 163 83 61 61 61 61 61
134 58 28 24 23 36 147 24 27 47
0 2 2 2 2 2 2 2 1 1
```

j. Realizar un subprograma que construya un histograma que represente la longitud de las palabras que se encuentran en un archivo texto. El programa debe tomar cada una de las palabras y por cada letra debe imprimir en otro archivo texto un asterisco, simulando una barra del histograma. **Entrada:** La primera línea del archivo de entrada contiene el número de barras que debe tener el histograma y las siguientes líneas contienen las palabras que deben representarse en el mismo. **Salida:** La salida deseada es un archivo texto con el histograma antes mencionado, el cual debe tener barras representadas por asteriscos correspondientes al número de letras que tiene cada palabra del archivo de entrada.

Ejemplo de archivo de entrada:

```
4
Prueba
Programación
Hola
Sistemas
```

Ejemplo de archivo de salida:

```
 *
 *
 *
 *
 * *
 * *
** *
** *
****
****
****
****
```

k – Se necesita realizar un programa que compare distintos valores que pueden tener diferentes bases (en nuestro caso sólo serán base binaria o base decimal). Para ello se da un archivo texto con un número indeterminado de líneas, donde en cada línea habrán cuatro números enteros (a,b,c,d), donde a y c representan los dos números a comparar y b y d serán sus respectivas bases.
Se desea producir un el archivo de salida en el cual por cada línea del archivo de entrada se escriba el carácter '>', '<' o '=', dependiendo si "a" es mayor, menor o igual a "c" (tomando en cuenta sus bases).

Ejemplo de archivo de entrada:

```
10 10 12 10
1011 2 9 10
12 10 1100 2
101 2 101 2
1001 10 1001 2
```

Ejemplo de archivo de salida:

```
<
>
=
=
>
```

l - Un tautograma es un poema, verso u oración formado por palabras que empiezan por la misma letra. Por ejemplo, las siguientes oraciones son tautogramas:

- Cristina come ciruelas con Carmen Castillejo cuando cantan canciones cubanas
- Pedro Pablo, pobre pintor, pinta preciosos paisajes por poca plata

Dado un archivo texto con un número indeterminado de oraciones, cada una en una línea, realizar un subprograma que produzca un archivo texto donde cada línea tenga un "Si" o un "No" dependiendo si en el archivo original la oración que se encontraba en esa línea era o no un tautograma.

Ejemplo de archivo de entrada:

```
Antes alegre andaba; agora apenas
Para poder presumir practica previamente
Procurad pensar poco, pero profundamente
Esto NO es un tautograma
Bebe Benito brandy. Bebe buen borracho
```

Ejemplo de archivo de salida:

```
Si
Si
Si
No
Si
```

m. Diseñe un subprograma que dado un archivo texto con un número indeterminado de palabras escritas una debajo de la otra, tenga como una salida una matriz NxN que sea una *sopa de letras* que contenga las palabras que se encuentran en el archivo texto. Las palabras deben aparecer aleatoriamente y con la misma probabilidad de ocurrencia en cualquier posición de la matriz y con alguna de las siguientes direcciones: horizontal (izquierda-derecha) o vertical (arriba-abajo). Asuma que todas las palabras pueden ser colocadas en la matriz y recuerde que las palabras se pueden cruzar. Los datos de entrada al subprograma son N y la dirección del archivo en el sistema.

n. Se desea enviar un mensaje de la siguiente forma: para cada palabra W se selecciona un número n y se replica W n veces horizontalmente. Luego la cadena de caracteres se repite en la siguiente línea, pero esta vez rotando los caracteres una vez a la izquierda. Este proceso se repite hasta que la palabra W aparezca escrita en la primera columna del patrón rectangular producido por este procedimiento.
Por ejemplo, cuando se escoge la palabra Hola y el número 3, debemos producir el siguiente patrón:

```
HolaHolaHola
olaHolaHolaH
laHolaHolaHo
aHolaHolaHol
```

Como entrada se da la dirección de un archivo texto que contiene información de una cantidad indeterminada de palabras en líneas separadas antecedidas por un número entero menor a 10.
Como salida debe generarse un archivo texto codificado de la forma anteriormente descrita.

Ejemplo de entrada:

```
2 Delphi
4 Quiz
```

Ejemplo de salida:

```
DelphiDelphi
elphiDelphiD
lphiDelphiDe
phiDelphiDel
hiDelphiDelp
iDelphiDelph
QuizQuizQuizQuiz
uizQuizQuizQuizQ
izQuizQuizQuizQu
zQuizQuizQuizQui
```

o. Una pareja de novios se dan cita cada día en la terraza de un café al término de su jornada laboral. Ambos llegan entre las 5:00 p.m. y las 5:45 p.m., de manera equiprobable e independiente. Permanecen allí durante T minutos. Diseñe una función que permita conocer la probabilidad de que la pareja se encuentre en dicho café.