EJERCICIOS PARA PREPARACIÓN MARATON DE PROGRAMACIÓN

ANGELA BIBIANA ORTEGON FUENTES

ANALISIS Y DESARROLLO DE SOFTWARE – SENA

NIVEL BASICO

1. **Sumatoria Simple:**
   Dado un número entero n, calcula la sumatoria de los números desde 1 hasta n.
   - **Entrada:** Un entero n.
   - **Salida:** La suma de los números del 1 al n.

```
ALGORITMO SumatoriaSimple
  LEER n
  suma ← 0
  PARA i ← 1 HASTA n HACER
    suma ← suma + i
  FIN PARA
  IMPRIMIR suma
FIN ALGORITMO
```

2. **Palíndromo:**
   Comprueba si una palabra o número es un palíndromo.
   - **Entrada:** Una cadena de texto.
   - **Salida:** "Sí" si es un palíndromo, "No" en caso contrario.

```
ALGORITMO Es Palíndromo
  LEER palabra
  SI palabra == REVERSO(palabra) ENTONCES
    IMPRIMIR "Sí"
```

```
  SINO
    IMPRIMIR "No"
  FIN SI
FIN ALGORITMO
```

3. **Múltiplos de 3 y 5:**
   Encuentra la suma de todos los números menores a n que sean múltiplos de 3 o 5.

   - **Entrada:** Un entero n.
   - **Salida:** La suma de los múltiplos.

```
ALGORITMO SumaMultiplos
  LEER n
  suma ← 0
  PARA i ← 1 HASTA n-1 HACER
    SI i MOD 3 == 0 O i MOD 5 == 0 ENTONCES
      suma ← suma + i
    FIN SI
  FIN PARA
  IMPRIMIR suma
FIN ALGORITMO
```

Nivel Intermedio

1. **Ordenar palabras por longitud:**
   Dado un conjunto de palabras, ordénalas por longitud.
   - **Entrada:** Una lista de palabras separadas por espacios.
   - **Salida:** Las palabras ordenadas por longitud, en caso de empate, por orden alfabético.

```
ALGORITMO OrdenarPalabras
  LEER palabras
  lista ← palabras SEPARAR POR ESPACIOS
```

```
  ORDENAR lista POR longitud, ALFABÉTICO EN CASO DE EMPATE
  IMPRIMIR lista
FIN ALGORITMO
```

2. **Juego de números:**
   Un jugador escoge un número xxx. Tú debes determinar si es posible formar xxx usando la suma de dos números de una lista.
   - **Entrada:** Un número xxx y una lista de números.
   - **Salida:** "Sí" o "No".

```
ALGORITMO SumaEnLista
  LEER x, lista
  PARA cada num1 EN lista HACER
    PARA cada num2 EN lista HACER
      SI num1 + num2 == x ENTONCES
        IMPRIMIR "Sí"
        SALIR
      FIN SI
    FIN PARA
  FIN PARA
  IMPRIMIR "No"
FIN ALGORITMO
```

3. **Conversión de bases:**
   Convierte un número de base decimal a base bbb.
   - **Entrada:** Un número en base 10 y una base bbb ($2 \leq bbb \leq 16$).
   - **Salida:** El número convertido a la base bbb.

```
ALGORITMO ConvertirBase
  LEER num, base
  resultado ← ""
  MIENTRAS num > 0 HACER
    resto ← num MOD base
```

```
    resultado ← CONCATENAR(ConvertirSimbolo(resto), resultado)
    num ← num DIV base
  FIN MIENTRAS
  IMPRIMIR resultado
FIN ALGORITMO
```

Nivel Avanzado

1. **Camino más corto (Grafos):**
   Dado un grafo ponderado representado como lista de adyacencia, encuentra el camino más corto desde un nodo origen hasta un nodo destino utilizando Dijkstra.
   - **Entrada:** Número de nodos, aristas y la lista de adyacencias.
   - **Salida:** La distancia mínima.

```
ALGORITMO Dijkstra
  LEER grafo, origen, destino
  distancias ← INFINITO PARA TODOS LOS NODOS
  distancias[origen] ← 0
  cola ← NODO_ORIGEN
  MIENTRAS cola NO ESTÉ VACÍA HACER
    nodo_actual ← EXTRAER_MINIMO(cola)
    PARA cada vecino EN grafo[nodo_actual] HACER
      distancia ← distancias[nodo_actual] + peso(nodo_actual, vecino)
      SI distancia < distancias[vecino] ENTONCES
        distancias[vecino] ← distancia
        ACTUALIZAR cola CON vecino
      FIN SI
    FIN PARA
  FIN MIENTRAS
  IMPRIMIR distancias[destino]
FIN ALGORITMO
```

2. **Subsecuencia más larga común:**
   Calcula la subsecuencia más larga común entre dos cadenas.
   - **Entrada:** Dos cadenas de texto.
   - **Salida:** La longitud de la subsecuencia más larga común.

```
ALGORITMO LCS
  LEER cadena1, cadena2
  m ← LONGITUD(cadena1)
  n ← LONGITUD(cadena2)
  matriz ← MATRIZ(m+1, n+1) INICIALIZADA EN 0
  PARA i ← 1 HASTA m HACER
    PARA j ← 1 HASTA n HACER
      SI cadena1[i] == cadena2[j] ENTONCES
        matriz[i][j] ← matriz[i-1][j-1] + 1
      SINO
        matriz[i][j] ← MAX(matriz[i-1][j], matriz[i][j-1])
      FIN SI
    FIN PARA
  FIN PARA
  IMPRIMIR matriz[m][n]
FIN ALGORITMO
```

3. **Coloreo de grafos:**
   Dado un grafo, determina el número mínimo de colores necesarios para colorearlo sin que dos nodos adyacentes compartan el mismo color.
   - **Entrada:** Número de nodos, aristas y la lista de adyacencias.
   - **Salida:** El número mínimo de colores.

```
ALGORITMO ColorearGrafo
  LEER grafo, num_nodos
  colores ← ARREGLO(num_nodos) INICIALIZADO EN -1
  PARA nodo ← 1 HASTA num_nodos HACER
```

```
    colores_usados ← COLORES_DE_VECINOS(grafo, nodo)
    color ← PRIMER_COLOR_DISPONIBLE(colores_usados)
    colores[nodo] ← color
  FIN PARA
  IMPRIMIR MAX(colores) + 1
FIN ALGORITMO
```

Ejercicios con Funciones

1. **Cálculo de factorial con función recursiva:**

- Escribe una función factorial(n) que calcule el factorial de un número usando recursividad.
- **Entrada:** Un entero n.
- **Salida:** El factorial de n.

```
FUNCION Factorial(n)
  SI n == 0 O n == 1 ENTONCES
    RETORNAR 1
  SINO
    RETORNAR n * Factorial(n - 1)
  FIN SI
FIN FUNCION
```

2. **Números primos en un rango:**

- Crea una función es_primo(n) para verificar si un número es primo. Luego, usa esa función para encontrar todos los primos en un rango dado.
- **Entrada:** Dos enteros a y b.
- **Salida:** Lista de números primos entre a y b.

```
FUNCION EsPrimo(n)
  SI n <= 1 ENTONCES
    RETORNAR FALSO
```

```
  FIN SI
  PARA i ← 2 HASTA RAIZ(n) HACER
    SI n MOD i == 0 ENTONCES
      RETORNAR FALSO
    FIN SI
  FIN PARA
  RETORNAR VERDADERO
FIN FUNCION

ALGORITMO PrimosEnRango
  LEER a, b
  PARA i ← a HASTA b HACER
    SI EsPrimo(i) ENTONCES
      IMPRIMIR i
    FIN SI
  FIN PARA
FIN ALGORITMO
```

3. **Área y perímetro de un círculo:**

- Implementa dos funciones: calcular_area(radio) y calcular_perimetro(radio) para un círculo.
- **Entrada:** Radio del círculo.
- **Salida:** El área y el perímetro.

```
FUNCION CalcularArea(radio)
  RETORNAR PI * radio^2
FIN FUNCION

FUNCION CalcularPerimetro(radio)
  RETORNAR 2 * PI * radio
```

```
FIN FUNCION


ALGORITMO Circulo
  LEER radio
  area ← CalcularArea(radio)
  perimetro ← CalcularPerimetro(radio)
  IMPRIMIR "Área:", area
  IMPRIMIR "Perímetro:", perimetro
FIN ALGORITMO
```

Ejercicios con Matrices

1. **Suma de dos matrices:**
    - Escribe un programa que tome dos matrices del mismo tamaño y devuelva su suma.
    - **Entrada:** Dos matrices A y B.
    - **Salida:** Una matriz C, donde C[i][j]=A[i][j]+B[i][j]C[i][j] = A[i][j] + B[i][j]C[i][j]=A[i][j]+B[i][j].

```
ALGORITMO SumaMatrices
  LEER matrizA, matrizB
  FILAS ← LONGITUD(matrizA)
  COLUMNAS ← LONGITUD(matrizA[0])
  matrizC ← MATRIZ(FILAS, COLUMNAS)

  PARA i ← 1 HASTA FILAS HACER
    PARA j ← 1 HASTA COLUMNAS HACER
      matrizC[i][j] ← matrizA[i][j] + matrizB[i][j]
    FIN PARA
  FIN PARA
```

```
  IMPRIMIR matrizC
FIN ALGORITMO
```

2. **Matriz transpuesta:**
    - Dado una matriz MMM, genera su transpuesta TTT.
    - **Entrada:** Una matriz MMM.
    - **Salida:** Matriz TTT, donde T[i][j]=M[j][i]T[i][j] = M[j][i]T[i][j]=M[j][i].

```
ALGORITMO TransponerMatriz
  LEER matriz
  FILAS ← LONGITUD(matriz)
  COLUMNAS ← LONGITUD(matriz[0])
  transpuesta ← MATRIZ(COLUMNAS, FILAS)

  PARA i ← 1 HASTA FILAS HACER
    PARA j ← 1 HASTA COLUMNAS HACER
      transpuesta[j][i] ← matriz[i][j]
    FIN PARA
  FIN PARA

  IMPRIMIR transpuesta
FIN ALGORITMO
```

3. **Producto de matrices:**
    - Implementa un programa que calcule el producto de dos matrices AAA y BBB.
    - **Entrada:** Dos matrices compatibles para la multiplicación.
    - **Salida:** Matriz producto CCC.

```
ALGORITMO ProductoMatrices
  LEER matrizA, matrizB
  FILAS_A ← LONGITUD(matrizA)
```

```
  COLUMNAS_A ← LONGITUD(matrizA[0])
  COLUMNAS_B ← LONGITUD(matrizB[0])
  matrizC ← MATRIZ(FILAS_A, COLUMNAS_B)

  PARA i ← 1 HASTA FILAS_A HACER
    PARA j ← 1 HASTA COLUMNAS_B HACER
      matrizC[i][j] ← 0
      PARA k ← 1 HASTA COLUMNAS_A HACER
        matrizC[i][j] ← matrizC[i][j] + matrizA[i][k] * matrizB[k][j]
      FIN PARA
    FIN PARA
  FIN PARA

  IMPRIMIR matrizC
FIN ALGORITMO
```

Ejercicios con Vectores

1. **Máximo y mínimo en un vector:**
    - Encuentra el valor máximo y el mínimo de un vector.
    - **Entrada:** Un vector de números.
    - **Salida:** El valor máximo y mínimo.

```
ALGORITMO MaximoMinimo
  LEER vector
  maximo ← vector[0]
  minimo ← vector[0]

  PARA cada elemento EN vector HACER
    SI elemento > maximo ENTONCES
      maximo ← elemento
```

```
    FIN SI
    SI elemento < minimo ENTONCES
      minimo ← elemento
    FIN SI
  FIN PARA

  IMPRIMIR "Máximo:", maximo
  IMPRIMIR "Mínimo:", minimo
FIN ALGORITMO
```

2. **Producto escalar de dos vectores:**
    - Calcula el producto escalar de dos vectores AAA y BBB.
    - **Entrada:** Dos vectores del mismo tamaño.
    - **Salida:** El producto escalar.

```
ALGORITMO ProductoEscalar
  LEER vectorA, vectorB
  producto ← 0

  PARA i ← 1 HASTA LONGITUD(vectorA) HACER
    producto ← producto + (vectorA[i] * vectorB[i])
  FIN PARA

  IMPRIMIR producto
FIN ALGORITMO
```

3. **Ordenar un vector:**
    - Escribe un programa que ordene un vector de menor a mayor utilizando el algoritmo de burbuja (Bubble Sort).
    - **Entrada:** Un vector de números.
    - **Salida:** El vector ordenado.

```
ALGORITMO OrdenarBurbuja
  LEER vector
  n ← LONGITUD(vector)

  PARA i ← 1 HASTA n-1 HACER
    PARA j ← 1 HASTA n-i HACER
      SI vector[j] > vector[j+1] ENTONCES
        TEMP ← vector[j]
        vector[j] ← vector[j+1]
        vector[j+1] ← TEMP
      FIN SI
    FIN PARA
  FIN PARA

  IMPRIMIR vector
FIN ALGORITMO
```

Ejercicios Combinados

1. **Rotación de una matriz:**
    - Escribe un programa que rote una matriz n×nn \times nn×n 90 grados hacia la derecha.
    - **Entrada:** Una matriz cuadrada.
    - **Salida:** La matriz rotada.

```
ALGORITMO RotarMatriz
  LEER matriz
  n ← LONGITUD(matriz)
  matriz_rotada ← MATRIZ(n, n)

  PARA i ← 1 HASTA n HACER
```

```
    PARA j ← 1 HASTA n HACER
      matriz_rotada[j][n-i+1] ← matriz[i][j]
    FIN PARA
  FIN PARA

  IMPRIMIR matriz_rotada
FIN ALGORITMO
```

2. **Vector de sumas por filas:**
    - Calcula un vector donde cada elemento es la suma de los valores de cada fila de una matriz.
    - **Entrada:** Una matriz MMM.
    - **Salida:** Un vector donde el elemento iii es la suma de la fila iii de MMM.

```
ALGORITMO VectorSumaPorFilas
  LEER matriz
  FILAS ← LONGITUD(matriz)
  COLUMNAS ← LONGITUD(matriz[0])
  vector_suma ← VECTOR(FILAS)

  PARA i ← 1 HASTA FILAS HACER
    suma ← 0
    PARA j ← 1 HASTA COLUMNAS HACER
      suma ← suma + matriz[i][j]
    FIN PARA
    vector_suma[i] ← suma
  FIN PARA

  IMPRIMIR vector_suma
FIN ALGORITMO
```

3. **Multiplicación de una matriz por un vector:**
    - Dado una matriz M y un vector V, calcula el producto resultante.
    - **Entrada:** Una matriz M y un vector V.
    - **Salida:** Un vector resultante.

```
ALGORITMO MultiplicarMatrizPorVector
  LEER matriz, vector
  FILAS ← LONGITUD(matriz)
  COLUMNAS ← LONGITUD(matriz[0])
  LONGITUD_VECTOR ← LONGITUD(vector)

  SI COLUMNAS != LONGITUD_VECTOR ENTONCES
    IMPRIMIR "Error: Dimensiones incompatibles"
    TERMINAR
  FIN SI

  vector_resultante ← VECTOR(FILAS)

  PARA i ← 1 HASTA FILAS HACER
    suma ← 0
    PARA j ← 1 HASTA COLUMNAS HACER
      suma ← suma + (matriz[i][j] * vector[j])
    FIN PARA
    vector_resultante[i] ← suma
  FIN PARA

  IMPRIMIR vector_resultante
FIN ALGORITMO
```

EJERCICIOS VARIADOS

1. Un grupo de *n* personas quiere cruzar un puente por la noche, solo pueden cruzar como mucho dos personas a la vez, cada grupo debe llevar una lámpara; pero solo hay una por lo que deben organizarse de tal forma que después que crucen alguien tiene que devolverse con la lámpara.

Cada persona tiene una velocidad diferente por lo que la velocidad del par que crucen es la velocidad del más lento.

La tarea consiste en idear un procedimiento que permita cruzar al grupo de personas en el menor tiempo posible.

Entrada: El número de personas a cruzar y las velocidades de cada persona

Salida: La suma total del tiempo y la descripción de la cruzada del puente indicando las personas o persona identificadas por sus tiempos al ir y/o volver.

```
ALGORITMO CruzarPuente

   LEER n       // Número de personas

   LEER velocidades[1..n] // Velocidades de cada persona

   ORDENAR velocidades EN ORDEN ASCENDENTE


   tiempo_total ← 0

   DESCRIPCION_CRUCE ← LISTA VACIA


   MIENTRAS LONGITUD(velocidades) > 3 HACER

      // Caso 1: Envío de los dos más lentos

      tiempo_estrategia_1 ← velocidades[1] + 2*velocidades[2] +
velocidades[LONGITUD(velocidades)]

      // Caso 2: Envío del más rápido con el más lento

      tiempo_estrategia_2 ← 2*velocidades[1] +
velocidades[LONGITUD(velocidades)-1] +
velocidades[LONGITUD(velocidades)]


      SI tiempo_estrategia_1 <= tiempo_estrategia_2 ENTONCES

         // Estrategia 1:
```

```
        AÑADIR ("Ir:", velocidades[1], velocidades[2]) A
DESCRIPCION_CRUCE
        AÑADIR ("Volver:", velocidades[1]) A DESCRIPCION_CRUCE
        AÑADIR ("Ir:", velocidades[LONGITUD(velocidades)-1],
velocidades[LONGITUD(velocidades)]) A DESCRIPCION_CRUCE
        AÑADIR ("Volver:", velocidades[2]) A DESCRIPCION_CRUCE
        tiempo_total ← tiempo_total + tiempo_estrategia_1
    SINO
        // Estrategia 2:
        AÑADIR ("Ir:", velocidades[1],
velocidades[LONGITUD(velocidades)]) A DESCRIPCION_CRUCE
        AÑADIR ("Volver:", velocidades[1]) A DESCRIPCION_CRUCE
        AÑADIR ("Ir:", velocidades[1],
velocidades[LONGITUD(velocidades)-1]) A DESCRIPCION_CRUCE
        AÑADIR ("Volver:", velocidades[1]) A DESCRIPCION_CRUCE
        tiempo_total ← tiempo_total + tiempo_estrategia_2
    FIN SI

    ELIMINAR LOS DOS MÁS LENTOS DE velocidades
  FIN MIENTRAS

  // Manejo del caso final con 3 personas o menos
  SI LONGITUD(velocidades) = 3 ENTONCES
    tiempo_total ← tiempo_total + velocidades[1] + velocidades[2] +
velocidades[3]
    AÑADIR ("Ir:", velocidades[1], velocidades[2]) A
DESCRIPCION_CRUCE
    AÑADIR ("Volver:", velocidades[1]) A DESCRIPCION_CRUCE
    AÑADIR ("Ir:", velocidades[1], velocidades[3]) A
DESCRIPCION_CRUCE
  SINO SI LONGITUD(velocidades) = 2 ENTONCES
```

tiempo_total ← tiempo_total + velocidades[2]

AÑADIR ("Ir:", velocidades[1], velocidades[2]) A DESCRIPCION_CRUCE

SINO SI LONGITUD(velocidades) = 1 ENTONCES

tiempo_total ← tiempo_total + velocidades[1]

AÑADIR ("Ir:", velocidades[1]) A DESCRIPCION_CRUCE

FIN SI

// Salida final

IMPRIMIR "Tiempo total:", tiempo_total

IMPRIMIR "Descripción del cruce:"

PARA cada paso EN DESCRIPCION_CRUCE HACER

IMPRIMIR paso

FIN PARA

FIN ALGORITMO

2. Dada una lista de números enteros positivos que puede contener desde 5 hasta 1000 elementos, por ejemplo (4, 6, 1, 90, 56, 43, 79, 23), se desea saber cuáles dos de ellos son los más cercanos en valor, es decir, que el valor absoluto de su diferencia sea menor que el de cualquier otra pareja de números en la lista.

Para la lista anterior, la pareja más cercana son 4 y 6, ya que ningún otro par de números tiene una resta cuyo valor absoluto sea menor a 2 (6-4=2).

Escribe de la manera más eficiente posible, un algoritmo que permita, mediante el uso de una computadora obtener la pareja más cercana de una lista dada.

Entrada: Lista de números.
Salida: Pareja más cercana.

ALGORITMO ParejaMasCercana

LEER lista

ORDENAR lista EN ORDEN ASCENDENTE

```
  diferencia_minima ← INFINITO
  pareja_mas_cercana ← []

  PARA i ← 1 HASTA LONGITUD(lista) - 1 HACER
    diferencia ← | lista[i+1] - lista[i] |
    SI diferencia < diferencia_minima ENTONCES
      diferencia_minima ← diferencia
      pareja_mas_cercana ← [lista[i], lista[i+1]]
    FIN SI
  FIN PARA

  IMPRIMIR "Pareja más cercana:", pareja_mas_cercana
  IMPRIMIR "Diferencia mínima:", diferencia_minima
FIN ALGORITMO
```

3. Dados dos enteros m y n, escriba un programa que construya una matriz con m renglones y n columnas cuyas entradas sean los números 1, 2,..., m*n acomodados en espiral, comenzando con el número 1 en la entrada que está en la esquina superior izquierda, siguiendo hacia la derecha, luego hacia abajo, luego hacia la izquierda, luego hacia arriba, y así sucesivamente.

Entrada: Dos números enteros m y n, cuyos valores están entre 1 y 100 (incluyéndolos).
Salida: Matriz.

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

```
ALGORITMO MatrizEspiral
  LEER m, n // Dimensiones de la matriz
```

```
CREAR matriz[m][n] INICIALIZADA EN 0

// Inicializar límites de los bordes
limite_superior ← 0
limite_inferior ← m - 1
limite_izquierdo ← 0
limite_derecho ← n - 1

numero_actual ← 1 // Primer número a insertar

MIENTRAS numero_actual <= m * n HACER
  // Llenar hacia la derecha
  PARA col ← limite_izquierdo HASTA limite_derecho HACER
    matriz[limite_superior][col] ← numero_actual
    numero_actual ← numero_actual + 1
  FIN PARA
  limite_superior ← limite_superior + 1

  // Llenar hacia abajo
  PARA fila ← limite_superior HASTA limite_inferior HACER
    matriz[fila][limite_derecho] ← numero_actual
    numero_actual ← numero_actual + 1
  FIN PARA
  limite_derecho ← limite_derecho - 1

  // Llenar hacia la izquierda
  SI limite_superior <= limite_inferior ENTONCES
    PARA col ← limite_derecho HASTA limite_izquierdo PASO -1 HACER
```

```
        matriz[limite_inferior][col] ← numero_actual
        numero_actual ← numero_actual + 1
      FIN PARA
      limite_inferior ← limite_inferior - 1
    FIN SI

    // Llenar hacia arriba
    SI limite_izquierdo <= limite_derecho ENTONCES
      PARA fila ← limite_inferior HASTA limite_superior PASO -1 HACER
        matriz[fila][limite_izquierdo] ← numero_actual
        numero_actual ← numero_actual + 1
      FIN PARA
      limite_izquierdo ← limite_izquierdo + 1
    FIN SI
  FIN MIENTRAS

  // Imprimir la matriz
  PARA i ← 0 HASTA m - 1 HACER
    PARA j ← 0 HASTA n - 1 HACER
      IMPRIMIR matriz[i][j], " "
    FIN PARA
    IMPRIMIR NUEVA_LINEA
  FIN PARA
FIN ALGORITMO
```

4. Desarrolle un algoritmo que lea el sueldo básico de n empleados y calcule el neto a pagar, si al sueldo básico se le suma una bonificación de 10% si el básico es menor o igual a 1.000.000 o del 5% en caso contrario.

Entrada: La primera línea indica el número de empleados. Las siguientes líneas el sueldo de cada empleado.

Salida: Corresponde al salario neto de cada uno de los empleados.

Ejemplo del archivo de entrada.
3
980000
1500000
1000000

Ejemplo del archivo de salida

Sueldo neto empleado 1: 1078000
Sueldo neto empleado 2: 1575000
Sueldo neto empleado 3: 1100000

```
Inicio
   Leer numero_empleados
   Para cada empleado desde 1 hasta numero_empleados hacer
      Leer sueldo_basico
      Si sueldo_basico <= 1000000 entonces
         bono = 10% del sueldo_basico
      Sino
         bono = 5% del sueldo_basico
      FinSi
      sueldo_neto = sueldo_basico + bono
      Mostrar "Sueldo neto empleado i: sueldo_neto"
   FinPara
Fin
```

5. Crear un programa que lea un número entero y a partir de él cree un cuadrado de asteriscos con ese tamaño. Los asteriscos sólo se verán en el borde del cuadrado, no en el interior.

Entrada: La primera línea del archivo contiene un número que indica el número de asteriscos que tendrá cada lado del cuadrado (n).

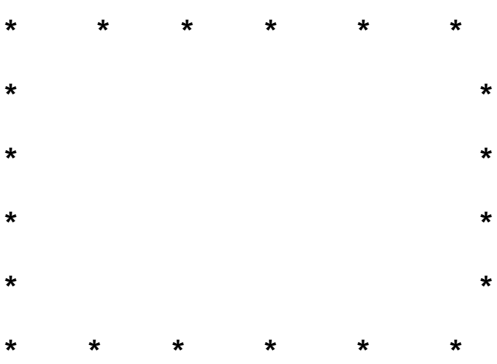
Salida: De acuerdo al número introducido por el usuario debe aparecer en pantalla un cuadro de longitud n asteriscos.

Ejemplo del archivo de entrada

6

Ejemplo del archivo de salida

6



Inicio

Leer n // Tamaño del cuadrado (número de asteriscos en cada lado)

Si n es mayor que 1

Imprimir una línea de asteriscos de tamaño n (esto es la primera línea)

Para i desde 1 hasta n - 2 hacer

Imprimir un asterisco, luego (n - 2) espacios en blanco, luego otro asterisco (esto es una línea intermedia)

Fin Para

Imprimir una línea de asteriscos de tamaño n (esto es la última línea)

Fin Si

Fin

6. Realice un programa que lea una lista de frases e imprima una matriz que contenga la misma frase con un carácter corrido hacia la izquierda.

Entrada: La primera línea del archivo contiene un número entero que indica el número de frases contenidas, las siguientes líneas contienen una frase. Las frases no superan los 100 caracteres.

Salida: Por cada una de las frases de la entrada se debe crear una matriz que contenga la frase de entrada corrida en un carácter hacia la izquierda. El número de líneas de cada matriz será equivalente al número de caracteres de la frase. Cada una de las matrices debe ser separada por una línea en blanco.

Ejemplo del archivo de entrada

hola mundo

programa

Ejemplo del archivo de salida

hola mundo

ola mundoh

la mundoho

a mundohol

mundohola

mundohola

undohola m

ndohola mu

dohola mun

ohola mund

programa

rogramap

ogramapr

gramapro

ramaprog

amaprogr

maprogra

aprogram

Inicio

Leer el número de frases, n // El primer número es el número de frases

Para i desde 1 hasta n hacer

Leer la frase // Leer cada frase de entrada

Obtener la longitud de la frase, longitud // Calcular cuántos caracteres tiene la frase

Para j desde 0 hasta longitud - 1 hacer

Crear una nueva cadena que consiste en los caracteres de la frase, comenzando desde el índice j hasta el final

Concatenar los caracteres desde el inicio hasta el índice j - 1 para completar la frase "corrida"

Imprimir la nueva cadena (matriz de una línea)

Fin Para

Imprimir una línea en blanco // Separar las matrices de frases con una línea en blanco

Fin Para

Fin

7. Trabajas para los Laboratorios de Propulsión por Reacción Sputnik. En este momento es necesario que escribas un programa que lea una matriz, la cual contiene una representación digitalizada de una fotografía del cielo. Cada elemento de la matriz representa la cantidad de luz que existe en determinada región de la imagen digitalizada. El rango de intensidad va de 0 a 20. El programa permitirá localizar las regiones donde se ubica una estrella, partiendo de la siguiente información: Una estrella se encuentra en el área cubierta por el elemento i,j de la matriz si se cumple la siguiente condición:

(MD(i,j) + suma de intensidades circundantes)/5 > 10.0 Donde MD representa la matriz digitalizada.

Entrada: La primera línea son dos enteros menores que 10 que indican el número de filas y columnas de la matriz, las siguientes líneas contienen enteros que representan las intensidades de cada posición de la matriz, separados por un blanco. Salida La salida deseada es la matriz que contiene un asterisco en la posición donde está localizada una estrella, y un blanco donde no la hay. La matriz debe estar circundada por un borde que indique las coordenadas de cada estrella. La coordenada inicial es 1. Cada elemento de la matriz debe estar separado por un espacio en blanco.

Ejemplo del archivo de entrada

```
6 8
0 3 4 20 15 0 6 8
5 13 6 8 2 0 2 3
2 6 2 2 3 0 10 0
0 0 4 15 4 1 1 20
0 0 7 2 6 9 10 4
5 0 6 10 6 4 8 0
```

Ejemplo del archivo de salida

```
   1 2 3 4 5 6 7 8
1      * *         1
2      * *         2
3      *           3
4              *   4
5        * *   *   5
6                  6
   1 2 3 4 5 6 7 8
```

Inicio

Leer el número de filas y columnas de la matriz (m, n)

Crear una matriz de tamaño m x n para almacenar las intensidades de luz

Leer los valores de la matriz MD (intensidades) desde la entrada

Crear una matriz de resultado del mismo tamaño m x n para marcar las estrellas con asteriscos y los demás lugares con espacios

Para i desde 1 hasta m-2 hacer // Iterar sobre las filas (sin bordes)

Para j desde 1 hasta n-2 hacer // Iterar sobre las columnas (sin bordes)

Calcular la suma de las intensidades circundantes (vecinos de la celda (i, j))

Si la condición (MD(i, j) + suma de intensidades circundantes)/5 > 10.0 se cumple entonces

Marcar el lugar como estrella en la matriz de resultado (poner un asterisco)

Sino

Marcar el lugar como vacío en la matriz de resultado (poner un espacio en blanco)

Fin Si

Fin Para

Fin Para

Para cada fila de la matriz de resultado hacer

Imprimir los elementos de esa fila, separados por espacios

Fin Para

Fin

8. Samuel Morse inventó el telégrafo en el año 1832 y diseño un esquema de codificación para enviar los mensajes. En el sistema, conocido como "Código Morse" o "Clave Morse", los caracteres se representan mediante puntos y líneas, que corresponden a impulsos eléctricos que producen una señal acústica o luminosa de una cierta duración. Así, la letra 'A' se codifica como un punto y una línea: .- Mientras que la letra 'F' como dos puntos, una línea y un punto: ..-.

A	.-	M	--	Y	-.--	6	-....
B	-...	N	-.	Z	--..	7	--...
C	-.-.	O	---	Ä	.-.-	8	---..
D	-..	P	.--.	Ö	---.	9	----.
E	.	Q	--.-	Ü	..--	.	.-.-.-
F	..-.	R	.-.	Ch	----	,	--..--
G	--.	S	...	0	-----	?	..--..
H	....	T	-	1	.----	!	··--·
I	..	U	..-	2	..---	:	---...
J	.---	V	...-	3	...--	“	.-..-.
K	-.-	W	.--	4	....-	‘	.----.
L	.-..	X	-..-	5	.....	=	-...-

Escribe de la manera más eficiente, un algoritmo que permita, traducir una frase a clave Morce.

Ejemplo:

Entrada: III MARATON DE PROGRAMACION UNISANGIL 2015

Salida: ..... --.-.-..------. -… .--..-.-----..-..---.--.-…----. ..---…….--.--….-.. ..--------.----…..

Inicio

Crear un diccionario que mapea caracteres a su código Morse

Leer la frase de entrada

Inicializar una lista vacía para almacenar las traducciones en código Morse

Para cada carácter en la frase hacer:

Si el carácter es un espacio, agregar un espacio a la lista

Si el carácter está en el diccionario, agregar su código Morse a la lista

Unir los elementos de la lista con espacios entre ellos

Imprimir la cadena resultante

Fin