Maestría en Ciencias con Especialidad en Computación y Matemáticas Industriales
Proceso de admisión 2025
Centro de Investigación en Matemáticas (CIMAT), A.C.
Examen de programación
(Tiempo: 3 horas)

Nombre: _____

Fecha: _____

Instrucciones:

- Escriba lo más claro posible para que al digitalizar las respuestas sean legibles.
- Resuelva cada problema en hojas independientes.
- Escriba su nombre en cada hoja.
- No se debería de pasar más de 30 minutos en cada problema.
- Escriba el código lo más claramente posible y especifique el lenguaje/pseudo-código utilizado.
- Si requiere funciones estándar como determinar el máximo o mínimo de dos valores (max(a, b), min(a,b)), ordenar (sort), determinar el tamaño de una cadena de caracteres (strlen), obtener el valor absoluto (abs), raíz cuadrada (sqrt) o logaritmos (log) puede utilizarlas sin implementarlas. Será necesario que implemente cualquier función diferente a las mencionadas anteriormente.

Problema 1 [1.5 puntos]

El determinante de una matriz $\mathbf{A}$ de dimensión 3×3 se calcula de la siguiente forma:

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}.$$

Para una matriz $\mathbf{B}$ de dimensión 4×4, su determinante se calcula utilizando la siguiente expresión:

$$|\mathbf{B}| = \begin{vmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{vmatrix} = b_{11}|\mathbf{B}_{11}| - b_{12}|\mathbf{B}_{12}| + b_{13}|\mathbf{B}_{13}| - b_{14}|\mathbf{B}_{14}|,$$

donde $\mathbf{B}_{ij}$ es la submatriz obtenida eliminando la i-ésima fila y la j-ésima columna de la matriz $\mathbf{B}$.

Desarrolle una función que reciba una matriz $\mathbf{B}$ de dimensión 4×4 y regrese el valor de su determinante $|\mathbf{B}|$.

Ejemplo: Para

$$\mathbf{B} = \begin{bmatrix} 2 & 1 & 3 & 4 \\ 0 & -1 & 2 & 1 \\ 3 & 2 & 0 & 5 \\ -1 & 3 & 2 & 1 \end{bmatrix},$$

la función regresará $|\mathbf{B}| = 35$, dado que

$$|\mathbf{B}| = 2\begin{vmatrix} -1 & 2 & 1 \\ 2 & 0 & 5 \\ 3 & 2 & 1 \end{vmatrix} - \begin{vmatrix} 0 & 2 & 1 \\ 3 & 0 & 5 \\ -1 & 2 & 1 \end{vmatrix} + 3\begin{vmatrix} 0 & -1 & 1 \\ 3 & 2 & 5 \\ -1 & 3 & 1 \end{vmatrix} - 4\begin{vmatrix} 0 & -1 & 2 \\ 3 & 2 & 0 \\ -1 & 3 & 2 \end{vmatrix}.$$

Problema 2 [1.5 puntos]

Dado un arreglo $A = [a_1, a_2, \ldots, a_n]$ donde cada valor de a_i es un valor entero, se pretende encontrar un subarreglo contiguo $A' = [a_i, a_{i+1}, \ldots, a_j]$, para $1 \leq i \leq j \leq n$ de A, tal que la suma de $\sum_{k=i}^{k=j} a_k$ sea la más grande posible. Diseñe un algoritmo utilizando recursividad para que, dado el arreglo A, encuentre el subarreglo de A'.

Pista. Considere el elemento que se encuentra justo en la posición $a_{\lfloor n/2 \rfloor}$. El arreglo A' puede caer en tres posibles casos:

- Todos los elementos de A' están de la posición 1 a la posición $\lfloor n/2 \rfloor - 1$.
- Todos los elementos de A' están de la posición $\lfloor n/2 \rfloor + 1$ a la posición n.
- El arreglo A' contiene entre sus elementos al elemento $a_{\lfloor n/2 \rfloor}$.

Ejemplo:

Entrada: $A = [-4, 2, -6, 8, -2, 4, 2, -10, 8]$.
Salida: $A' = [8, -2, 4, 2]$.

Problema 3 [1.5 puntos]

Ordene de forma ascendente una estructura de datos Cola (estructura FIFO: first in first out, la cual se asume que ya está implementada) de tipo de dato entero. El ordenamiento debe realizarse sin utilizar estructuras de datos adicionales, sólo la estructura original. Recordando que las operaciones básicas de una estructura de cola son las siguientes: push (añadir), pop (eliminar), front (acceder al elemento al frente), empty (verificar si está vacía).

Ejemplo:

Entrada: push(30), push(-30), push(100), push(10), push(20), push(50), push(40).
Salida: -30 10 20 30 40 50 100.

Problema 4 [2 puntos]

En base a la Figura 1, la integral definida $I = \int_{x_1}^{x_2} f(x)\,dx$ de una función $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ se puede aproximar como

$$I \approx \frac{N_{\bigoplus}}{N_{\bigodot} + N_{\bigoplus}} A, \tag{1}$$

donde $A = a \times b$, $N_{\bigodot}$ y $N_{\bigoplus}$ son el número de puntos con las etiquetas $\bigodot$ y $\bigoplus$, respectivamente.

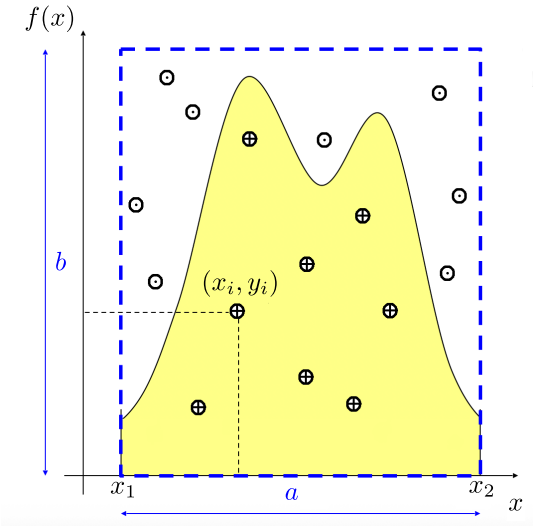

Figura 1: Aproximación de integral definida.

Considere que a usted se le proporciona una función `rand01`() que genera un número aleatorio $r \in \mathbb{R}$ en el intervalo $[0, 1]$. Desarrolle una función `ApproximateIntegral`(x_1, x_2, f, N) que haga uso de la Ecuación (1) para devolver la aproximación de I a partir de los límites de integración x_1 y x_2, la función $f(.)$, y un número entero $N = N_{\odot} + N_{\oplus}$ de puntos (x_i, y_i) a generarse aleatoriamente con la función `rand01`().

Problema 5 [1.5 puntos]

Dado un arreglo $A = [a_1, a_2, \ldots, a_n]$, donde todos los elementos a_i pueden tomar un valor en el conjunto $\{0, 1, \ldots, 2n\}$. Sea $M = \text{máx}\{a_i | a_i \in A\}$.

Genere un nuevo arreglo $C = [c_0, c_1, \ldots, c_M]$ en a lo más un número de pasos proporcional al valor de M, de forma tal que el valor en la j-ésima posición c_j contenga el número de veces que el elemento j se repite en A.

Ejemplo:

Entrada: $[8, 3, 4, 2, 1, 3, 2, 7]$.
Salida: $[0, 1, 2, 2, 1, 0, 0, 1, 1]$.

Problema 6 [2 puntos]

Diseñe un algoritmo que analice una secuencia de N números enteros y devuelva un arreglo de N salidas lógicas (booleanas). Cada valor en la salida indicará si el número actual es divisible entre alguno de los números anteriores en la secuencia observada hasta ese momento.

El algoritmo debe cumplir las condiciones ordenadas con prioridad:

- Prioridad 1: Mantener al mínimo la cantidad de memoria necesaria (cantidad total de elementos almacenados en memoria). Se busca que, en la mayoría de los casos, se almacenen menos de N elementos en memoria. Cuanto menor sea el uso de memoria es mejor.
- Prioridad 2: Minimizar el uso de la operación módulo ($\%$), ya que esta operación es más costosa que las operaciones de comparación (==, <=, >=, <, >).

Resuelva los siguientes puntos:

1. Escriba el código de tu algoritmo y explica brevemente cómo es que cumplen las condiciones de las prioridades 1 y 2 (0.75 puntos).

2. Muestre cómo funcionaría tu algoritmo con la secuencia de entrada S = [5, 3, 15, 2, 7, 14, 28, 4, 6, 12]. Debes incluir los datos que se almacenan en memoria (variables, listas, arreglos o cualquier estructura de almacenamiento que utilices) (0.25 puntos).
3. Escriba un caso de secuencia de entrada con 10 enteros en el que el número de veces que se utiliza la operación módulo sea el máximo posible para una secuencia de longitud 10. Explique por qué este caso representaría el peor caso en cuanto a tiempo de ejecución (0.25 puntos).
4. Escriba un caso de secuencia de entrada con 10 enteros en el que el número de veces que se utiliza la operación módulo sea el mínimo posible para una secuencia de longitud 10. Explique por qué este caso representaría el mejor caso en cuanto a tiempo de ejecución (0.25 puntos).
5. Comente cómo sería el uso de memoria en los dos casos anteriores. (0.25 puntos)