

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA Y ESTRUCTURA DE DATOS
Primer Examen
(Primer Semestre 2025)

Duración: 2h 50 min.

- **No puede utilizar apuntes, solo hojas sueltas en blanco.**
- En cada función el alumno deberá incluir, a modo de comentario, la forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- No puede emplear plantillas o funciones no vistas en los cursos de programación de la especialidad.
- Los programas deben ser desarrollados en el lenguaje C++. Si la implementación es diferente a la estrategia indicada o no la incluye, la pregunta no será corregida.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 50% del puntaje asignado a dicha pregunta.
- Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado. El orden será parte de la evaluación.
- **Solo está permitido acceder a la plataforma de PAIDEIA, cualquier tipo de navegación, búsqueda o uso de herramientas de comunicación se considera plagio por tal motivo se anulará la evaluación y se procederá con las medidas disciplinarias dispuestas por la FCI.**
- Para esta evaluación solo se permite el uso de las librerías `iostream`, `iomanip`, `string`, `cmath` o `fstream`
- Su trabajo deberá ser subido a PAIDEIA.
- **Es obligatorio usar como compilador NetBeans.**
- Los archivos deben llevar como nombre su código de la siguiente forma `codigo_EX1_P#&` (donde # representa el número de la pregunta a resolver y & representa la alternativa a o b)

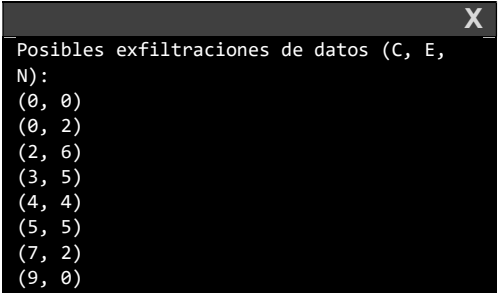
Elija 4 preguntas de las 5 que se muestran a continuación:

Pregunta 1 (10 puntos)

- a. **(5.0 puntos)** Durante una auditoría de ciberseguridad, se analiza la estructura lógica de una red de una empresa representada como una matriz de R filas por S columnas, donde cada celda corresponde a un nodo clasificado con un código: C (Nodo Crítico), E (Nodo Externo), N (Nodo en lista Negra), U (Nodo Usuario), A, B, X, Y, Z (otros tipos de nodos), y D (Nodo Desconectado). El objetivo es detectar posibles rutas de exfiltración de datos¹ que estarían compuestas por tres nodos consecutivos: Crítico, Externo y en Lista Negra (C, E y N) en cualquier orden. Estas rutas pueden alinearse horizontal, vertical o diagonalmente, y no deben contener nodos desconectados (D), ni considerar como válidas otras combinaciones.

Se debe implementar un programa iterativo que, utilizando la estrategia de **fuerza bruta**, recorra toda la matriz y detecte estas combinaciones válidas, imprimiendo las coordenadas (fila, columna) de la celda inicial de cada ruta. El análisis debe considerar las ocho direcciones posibles (horizontal, vertical y diagonales), sólo como combinaciones válidas C-E-N, y evitando analizar celdas desconectadas como punto de inicio.

¹ La exfiltración de datos es el proceso no autorizado mediante el cual información confidencial es transferida desde un sistema hacia un destino externo, comúnmente como resultado de ciberataques o accesos indebidos.

Entrada (directamente en el programa principal)	Salida
Matriz[R=12][S=8] <pre> {'C','E','N','Z','Y','X','A','U'}, {'D','Y','X','A','B','A','Y','U'}, {'A','Z','Y','A','X','B','C','U'}, {'B','Z','Z','A','E','E','Y','A'}, {'Y','Y','B','X','N','C','Z','Y'}, {'U','U','Y','Y','B','N','C','Z'}, {'Z','Y','A','A','U','Y','Y','E'}, {'Y','U','E','D','Z','B','B','N'}, {'Z','C','B','X','U','B','A','Z'}, {'N','Z','B','C','Y','Y','A','Y'}, {'X','A','E','Z','Z','A','U','Z'}, {'A','C','Z','X','X','Y','Y','Z'} </pre>	

- b. (5 puntos) En el almacén industrial **ColorPucp**, la mercancía se organiza sobre **palets de diferentes colores**, los cuales representan el tipo, la prioridad o la condición del producto almacenado. La codificación de colores es la siguiente:

- **Rojo (R):** productos inflamables
- **Amarillo (A):** productos perecibles
- **Azul (B):** productos no perecibles
- **Verde (V):** materiales reciclables o retornables

Esta codificación permite una **gestión visual y automatizada** del inventario.

El sistema cuenta con **robots autónomos**, representados por el carácter '**D**', que se ubican estratégicamente en el almacén (una matriz de 9x9). Su función principal es **retirar automáticamente los palets de un color específico**, ya sea por una necesidad operativa, una orden de reubicación o por un protocolo de seguridad.

Funcionamiento del Algoritmo de Extracción

Cuando se indica el color de palets a retirar (por ejemplo, azul), el robot analiza su entorno inmediato dentro del tablero (matriz de 9x9) y determina en qué dirección (horizontal, vertical o diagonal) se encuentra la mayor cantidad de palets del color solicitado. Si existe más de un mayor debe eliminar el primero que encuentre.

Se pide elaborar una función que a partir de la ubicación de la ficha '**D**' dada como parámetro permite eliminar las fichas en la dirección en dónde exista la mayor cantidad de fichas de un color dado también como parámetro. **Debe usar recursividad para contar palets y para eliminar palets.**

En el tablero que se da como ejemplo, si el palet por eliminar fuese el color azul(B), se eliminarían los palets azules de la dirección: Abajo.

Caso de uso:

Almacén antes:

	1	2	3	4	5	6	7	8	9
1
2	.	A	.	.	.	B	.	.	.
3	.	.	V	A
4	.	B	.	D	.	V	B	A	.
5	.	.	B	B
6	.	A	.	B
7	B	A	.	.
8	.	.	.	V
9	.	.	.	B	A

Ingrese el color de palets a retirar (R/A/B/V): B

Almacén después:

	1	2	3	4	5	6	7	8	9
1
2	.	A	.	.	.	B	.	.	.
3	.	.	V	A
4	.	B	.	D	.	V	B	A	.
5	.	.	B
6	.	A
7	B	A	.	.
8	.	.	.	V
9	A

"Este ejercicio está basado en un problema propuesto por Juan Guanira/PUCP en [Examen 1/Algoritmos y Estructura de Datos, 2004])."

Pregunta 2 (10 puntos)

- a. (5.0 puntos) Una fábrica maneja una lista simplemente enlazada de pedidos de producción, donde cada nodo representa un pedido con los siguientes campos: Código del pedido, Cantidad de unidades y Prioridad del pedido: 'A', 'B', 'C'

Los pedidos llegan en orden cronológico durante el día y se guardan en la lista. Al día siguiente, a primera hora, por un tema de logística, los pedidos deben agruparse bajo la siguiente lógica de reorganización:

1. Los pedidos de prioridad 'A' deben ir al inicio de la lista.
2. Luego, deben ir los grupos de pedidos del mismo tipo de prioridad (B o C).
 - 2.1. Cada grupo se define como una secuencia continua de 2 o más pedidos consecutivos con la misma prioridad (sin interrupciones de otra prioridad).
3. Dentro de cada grupo, el orden cronológico debe conservarse.
4. Finalmente, los pedidos que no pertenecen a ningún grupo (es decir, únicos o aislados) deben ir al final de la lista, en el orden en que aparecen.

Por ejemplo:

Un conjunto de pedidos recibidos en un día que se guardan en una lista en ese orden (Inicio a Fin).

Código	Cantidad	Prioridad
1	10	B
2	20	A
3	15	B
4	12	B
5	5	C
6	6	A
7	8	C
8	4	C
9	2	B

Al aplicar la reorganización, la lista de pedidos quedaría de la siguiente manera, en ese orden (Inicio a Fin):

Código	Cantidad	Prioridad
2	20	A
6	6	A
3	15	B
4	12	B
7	8	C
8	4	C
1	10	B
5	5	C
9	2	B

Se le pide:

- Implementar una función que reordene la lista siguiendo la lógica indicada. **No puede utilizar memoria extra de ningún tipo ni arreglos, vectores, estructuras auxiliares o archivos.** (4 puntos).
 - Implementar una función que imprima las listas en el formato indicado (1 punto).
- b. (5 puntos)** Imagine un sistema de monitoreo de seguridad para una máquina industrial que opera a altas temperaturas. Esta máquina genera una lectura de temperatura cada segundo. Por razones de seguridad, es fundamental calcular en todo momento el promedio de las **últimas K temperaturas** registradas, con el fin de detectar posibles sobrecalentamientos. Dado que el sistema debe operar **en tiempo real**, el procesamiento de cada nueva lectura **debe ser rápido y eficiente**. Además, por restricciones de memoria, **solo pueden almacenarse las últimas K lecturas** de temperatura.

Se le pide implementar un programa que reciba un número K (cuántas temperaturas se consideran en el promedio), y luego procese una secuencia de lecturas de temperatura, recibidas una por una desde la consola. Por cada temperatura ingresada, el programa debe mostrar en pantalla el **promedio de las últimas K temperaturas registradas** (o de las

disponibles si aún no se han recibido **K** valores). El proceso termina cuando se ingresa la temperatura -1.

Requisitos de implementación:

- Seleccione **la estructura de datos más adecuada** para almacenar sólo la últimas K temperaturas. Haga las adaptaciones necesarias.
- Implemente al menos dos funciones:
 - o **agregarTemperatura(temperatura)**: agregar una nueva lectura.
 - o **calcularPromedio()**: devuelve el promedio de las ultimas K temperaturas.
 - o **Ambas funciones deben ser eficientes (sin bucles ni estructuras iterativas).**

Ejemplo:

```
Ingrese K: 3
Ingrese Temperatura: 25
Promedio: 25
Ingrese Temperatura: 30
Promedio: 27.5
Ingrese Temperatura: 32
Promedio: 29
Ingrese Temperatura: 28
Promedio: 30
Ingrese Temperatura: 24
Promedio: 28
```

Pregunta 3 (5 puntos)

Una empresa dedicada al almacenamiento cuenta con un sistema basado en robots apiladores. Estas unidades son bastante básicas por tal motivo solo soportan el uso de pilas simples en su programación, en otras palabras, solo pueden guardar un número dentro de las pilas, el cual tiene 6 dígitos con el año y mes de producción unido (a este número se le llama lote), no pueden soportar más datos en los nodos de las pilas. Tampoco soportan variables globales de ningún tipo, pero si estructuras. Luego de un tiempo de operaciones se dan cuenta que muchas veces los robots han colocado lotes antiguos en la parte baja de las pilas o al medio lo cual complica la salida de los lotes más antiguos generando perdidas a la empresa. Por tal motivo le solicitan al área de TI que adapte los programas de manejo de pilas para que pueda generarse una función que obtenga el valor más antiguo de la pila. A continuación, algunos ejemplos:

Si se realizan las siguientes operaciones en una pila:

```
nuevoapilar(pila,201809)
nuevoapilar (pila, 202010)
nuevoapilar (pila, 202109)
nuevoapilar (pila, 201510);
nuevoapilar (pila, 202409);
```

Si se llama a la función que obtiene al más antiguo de la pila mostrará: 201510

Si luego se realizan las siguientes operaciones:

```
nuevodesapilar(pila) // debe devolver un dato  
nuevodesapilar(pila)
```

Si ahora se llama a la función que obtiene al más antiguo de la pila mostrará: 201809

Desarrolle un programa que utilizando que utilizando **dos pilas (una principal y una extra)** permita obtener el mínimo valor de la pila. Para que esta pregunta tenga validez debe crear las funciones de soporte necesarias: **apilar, desapilar, cima, espilavacia y minimo**. **No puede emplear variables globales, no puede realizar ninguna iterativa. Tampoco está permitido el uso de arreglos o tads adicionales a lo indicado. La pregunta es válida si desarrolla todas las funciones solicitadas, de faltar alguna no tiene valor su respuesta.**

Al finalizar el examen, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este examen.

Profesores del curso:

Ana Roncal
Fernando Huamán
David Allasi
Heider Sanchez
Rony Cueva

San Miguel, 24 de mayo del 2025