PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA

Primer Examen

(Primer Semestre 2020)

Duración: 3 horas

Nota:

- Está permitido el uso de apuntes de clase y librerías desarrolladas en clase.
- Debe utilizar comentarios para explicar la lógica seguida en los programas elaborados, si en caso no se indica como ejecutar el programa desarrollado, no se revisará el código desarrollado.
- No se califica código comentado.
- Si la implementación es significativamente 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.
- Las variables y funciones deben tener nombres apropiados, en caso contrario no se revisará el código desarrollado.
- El orden será parte de la evaluación.
- Debe usar tipos de datos adecuados, según la pregunta lo requiera.
- Su trabajo deberá ser subido a PAIDEIA en el espacio indicado por los jefes de práctica.

Puntaje total: 20 puntos.

Elija 2 de las siguientes preguntas:

Pregunta 1 (10 puntos)

Recordando que el sistema estándar GTIN-13 tiene un total de 13 dígitos, en el que los 3 primeros corresponden al código del país, los 4 siguientes al código de empresa, los 5 siguientes al código del producto y el último dígito corresponde al dígito de control que es calculado, se ha reservado el primer dígito del código del producto para la categoría del producto en la tienda. Las categorías de productos son: 1- Alimentos, 2-Aseo, 3-Electrónicos, 4-Bazar y 5-Juguetes. Si el código GTIN-13 de un producto es 7751234**5**67892, se puede ver que es un juguete porque el primer dígito del código de producto es 5.

Adicional al código GTIN-13, la tienda también utiliza un código extendido de 32 dígitos por producto para la gestión de inventario, al cual denomina trama. La trama de datos de este sistema tiene la siguiente estructura:

`<código_GTIN-13><precio_en_soles><stock><stock_mínimo><descuento>`

Donde:

- `código_GTIN-13`: corresponde a los 13 dígitos para el código GTIN-13 del producto.
- `precio_en_soles`: corresponde a los 7 dígitos para el precio original en soles del producto, considerando los dos últimos dígitos como la parte decimal.
- `stock`: corresponde a los 5 dígitos para el stock disponible del producto.
- `stock_mínimo`: corresponde a los 4 dígitos para el valor considerado como stock mínimo para ese producto.
- `descuento`: corresponde a los 3 dígitos para el porcentaje de descuento.

Como ejemplo, analicemos la siguiente trama:

`77512345678920039990000040005000`

La trama presentada contiene la siguiente información:

GTIN-13: 7751234**5**67892	Categ. del producto: 5 (Juguete)	Precio: 0039990 (S/. 399.90)
Stock: 00004 (4 unids.)	Stock mínimo: 0005 (5 unids.)	Dscto.: 000 (0% de dscto.)

Debido a la situación actual por el COVID-19, la tienda debe realizar la reposición diaria de los productos que se encuentren con un stock por debajo del stock mínimo para volver a tener en stock por lo menos 3 veces el número del stock mínimo; y solo en el caso de alimentos se debe contar con un stock de 20 veces el stock mínimo. Para ello, se requiere un programa que realice los pedidos de los productos que se obtienen secuencialmente de la base de datos representada por el conjunto de tramas de productos.
Es por ello que se necesita de una cola que permite ingresar los requerimientos de pedido de cada producto cuyo stock está por debajo del stock mínimo. Debido a que la tienda maneja un presupuesto límite, no se pueden concretar (comprar) todos los pedidos ingresados a la cola. Luego, el programa realiza el desencolado con prioridad de los pedidos de acuerdo con la categoría de cada producto para luego ingresarlos al sistema del proveedor y concretar la compra, recordando que la suma de todos los pedidos que se vayan a comprar no debe exceder el monto límite del presupuesto. Las prioridades para desencolar los pedidos de productos son: 1ro alimentos, 2do aseo, 3ro electrónicos, 4to bazar y 5to juguetes.
Se describe a continuación un caso de cómo se ingresan los pedidos a la cola y cómo se desencolan los pedidos para realizar la compra, considerando un valor ingresado por el administrador como el monto límite asignado (presupuesto) para el desembolso total por todas las compras de un día, y que, además, el monto a pagar por la compra de un producto se obtiene a partir del precio de cada producto, sabiendo que hay un margen de ganancia del 20% por la venta de cada producto.
Para este caso, la cantidad de productos (n) de la tienda es 10, entonces se cuenta con una base de datos de productos almacenados en un arreglo de tramas (matriz bidimensional de dígitos), 10 tramas de 32 dígitos cada uno.

```
/* Arreglo de tramas que representa los n=10 productos de la tienda */
Arr_trama = {77512344678920199990000020003000, 77512342234560009690000070008000,
77512343376540019990000100005000, 77512341969690001990000090010000,
77512343669990024990000020003000, 77512341696960000990000030010000,
77512342678920003990000040005000, 77512342876540002990000060005000,
77512343969690031990000020003000, 77512342669990002990000050008000}
```

Al recorrer las tramas de los productos de manera secuencial, se verifica, por cada uno, si el stock disponible es menor a su stock mínimo. De encontrar un producto que cumpla esa condición, se ingresa, a la cola de pedidos, la información de dicho producto. Considere como datos de un pedido (un nodo de la cola) los siguientes atributos: *código del producto*, la cantidad de productos a pedir y el precio de compra del producto en S/. Recuerde que la cantidad de unidades a pedir en el caso de alimentos es 20 veces el stock mínimo. Para obtener el precio de compra y así lograr un margen de ganancia del 20%, se divide el precio de venta entre 1.2. Y el monto a desembolsar por pedido es la cantidad multiplicada con el precio de compra. Entonces la cola de pedidos quedaría así:

Last Head

26699	*39696*	*26789*	*16969*	*36699*	*19696*	*22345*	*46789*
24	9	15	200	9	200	24	9
24.92	266.58	33.25	8.25	208.25	16.58	80.75	1666.58

El sistema comenzará a desencolar los pedidos de acuerdo con su prioridad, es decir, primero los alimentos (1), luego los de aseo (2), se pasará luego a los electrónicos (3), posteriormente los de bazar (4), y finalmente los juguetes (5). Antes de desencolar un pedido, se verificará si existe el presupuesto suficiente para la realizar la compra del pedido completo, caso contrario, no se desencola el pedido y solo se compraría la cantidad de unidades del producto que alcance con el presupuesto. Luego se actualizaría el valor de la cantidad de unidades del pedido por los que faltarían comprar y se actualizaría el monto asignado para el desembolso total. Cada vez que se desencola un pedido debe actualizarse el monto para el desembolso total, ya que el desencolar representa la confirmación de la compra del pedido, y es por ello que se reduce el monto presupuestado para las compras del día.
Si el valor del monto inicial ingresado por el administrador para el desembolso total por las compras es de S/. 11,000, entonces, en el primer intento de desencolar un pedido para realizar la compra se verifica si alcanza para comprar. Se recorre la cola y se busca el pedido que está

más adelante que corresponda con la categoría de alimentos. En este caso es el pedido que está en la 3ra posición contando a partir de la cabeza (*head*). Como el monto requerido para concretar el pago por el pedido es 200 x S/. 16.58 = S/. 3,316, entonces se requiere S/. 3,316 para realizar dicho pedido, y cómo alcanza el presupuesto, entonces se procede a desencolar y actualizar el monto del presupuesto a 11,000-3,316 = S/. 7,684, quedando la cola de la siguiente forma:

Last — Presupuesto: S/. 7,684 — Head

26699	39696	26789	16969	36699	22345	46789
24	9	15	200	9	24	9
24.92	266.58	33.25	8.25	208.25	80.75	1666.58

Luego, para los siguientes pedidos se repite el mismo procedimiento, actualizándose el monto del presupuesto por cada pedido desencolado. Se repite el procedimiento hasta llegar al siguiente estado de la cola antes de desencolar el siguiente pedido con un presupuesto de S/. 1,124.92:

Last	Head
39696	46789
9	9
266.58	1666.58

Al verificar si se puede realizar la compra del siguiente pedido de la cola, considerando que tiene mayor prioridad el producto de la categoría 3 al de la categoría 4, nos damos cuenta que se requiere 9 x S/. 266.58 = S/. 2,399.22 y solo se cuenta con S/. 1,124.92 en el presupuesto restante. Por ende, no se desencola el pedido, pero sí se compraría las unidades que permiten con ese monto restante y se actualizaría el valor de las unidades en el pedido de la cola. Con S/. 1,124.92 se pueden comprar 4 unidades de S/. 266.58 (S/. 1,066.32), quedando un resto en el monto del presupuesto de S/. 58.60. Finalmente, la cola quedaría así:

Last	Head
39696	46789
5	9
266.58	1666.58

Utilice el código fuente base disponible en **PAIDEIA** para este problema, en él encontrará los datos de las tramas con la información de los productos en un arreglo de tramas y las funciones base. Se le solicita lo siguiente:

a) (1.0 punto) Defina la estructura de datos que soporte la aplicación.

b) (2.0 puntos) Implemente una función que permita ingresar un pedido a la cola (encolar un pedido sin recorrer la cola) a partir de los datos de un pedido. Tenga en cuenta que el encolado no valida los datos del pedido, solo encola.

c) (4.0 puntos) Implemente una función que permita desencolar un pedido en base al presupuesto disponible e imprima los datos para la compra del pedido desencolado (código del producto, cantidad de unidades a comprar y monto a desembolsar). Considere, como un dato de salida de la función, el presupuesto que queda luego de ser utilizado para la compra de las unidades del pedido a desencolar. Para el desencolado, considere los criterios de prioridad explicados en el procedimiento descrito. En el caso que no se pueda desencolar porque el presupuesto no alcanza para el total de unidades solicitadas del producto que se debería desencolar, se procederá a actualizar la cantidad a pedir de dicho producto según lo explicado, consumiendo el presupuesto por las unidades que sí se pueden comprar; también debe imprimir los datos de las unidades que sí se comprarían de dicho producto. La función **desencolar** es la única que permite que se pueda recorrer la cola (trabaja como en una lista). Para solucionar esta pregunta **no puede utilizar memoria adicional o estructuras auxiliares**

d) (1.0 punto) Implemente una función que imprima, recorriendo la cola del último a la cabeza, los datos de los pedidos que se quedaron en la cola, si los hubiera. El formato de impresión sería:

```
Datos de los pedidos que quedaron en la cola:
   Prod.: 39696, cantidad a pedir: 5, precio unitario de compra: 266.58
   Prod.: 46789, cantidad a pedir: 9, precio unitario de compra: 1666.58
```

e) (2.0 puntos) Implemente el algoritmo en la función **main** que:

- Lea el valor del monto límite inicial (presupuesto) que ingresa el usuario administrador.
- Realice el ingreso de todos los pedidos a la cola a partir de los datos que se encuentran en el arreglo de tramas. Debe validar qué productos son los que se deben de pedir.
- Simule la realización de la compra de los pedidos de la cola (**desencolar** los pedidos ingresados) considerando lo que el presupuesto permite. Recuerde que por cada desencolado de pedido se actualiza e imprime el presupuesto.
- Muestre los datos de los pedidos que se quedaron en la cola.

Pregunta 2 (10 puntos)

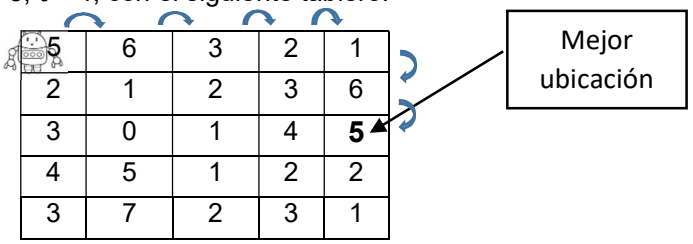
Una empresa de almacenamiento, ha adquirido un ***robot apilador***, que soporta programas en ANSI C, y su función será colocar pilas de cajas dentro del almacén que tiene n x n espacios. Se sabe que por temas mecánicos el robot solo puede moverse hacia la derecha o hacia abajo. Además para llevar un control adecuado, el almacén se representa como un tablero donde cada casillero es el espacio donde se puede colocar una pila de cajas. Por tal motivo el tablero n x n tendrá registrado en cada casillero, el tamaño de la pila adicional que puede recibir. Por ejemplo si un casillero permite apilar 7 cajas más, el tablero mostrará el valor de 7.

Las tareas que el robot debe realizar son las siguientes:

- El robot recibe una pila **p** de productos que debe almacenar, así como el tamaño de esta pila (**t**). Con esta información debe encontrar el espacio o casillero, donde puede ubicar la pila de tamaño **t**. **Para esta operación debe priorizar los espacios que se encuentren más lejanos al punto de partida (0,0)**. Para este cálculo, el robot lleva el control de sus pasos dentro del tablero. Se sabe que este autómata va de un casillero a otro en un solo paso. Recuerde: **t** <= tamaño de posición elegida.

Por ejemplo:

Para el ingreso de datos: n = 5, **t** = 4, con el siguiente tablero:



5	6	3	2	1
2	1	2	3	6
3	0	1	4	**5**
4	5	1	2	2
3	7	2	3	1

La respuesta será:

El mejor espacio está a **6** pasos de distancia y acepta pilas de tamaño **5**.

- La segunda tarea que realiza el ***robot apilador*** es unir la pila **u** que ya se encuentra en la posición elegida y la pila **p** que recibió para almacenar. Se sabe que la pila **u** y la pila **p**, se encuentran ordenadas por su código (mientras menor es el código más antiguo es el producto). Por tal motivo debe unir ambas pilas en la pila **u**, ordenada por código.

Por ejemplo:

Pila **u**:

2	5	6

Pila **p**:

1	3	4	5

Resultado:

Pila **u**:

1	2	3	4	5	5	6

Como especialista en algoritmia se le solicita:

a) Desarrolle el ingreso de datos para el **tablero**, la pila **p** y su tamaño **t** a ubicar. (0.5 punto)
b) Desarrollar una función recursiva para que el robot realice la primera tarea de buscar la posición más adecuada, solo se debe retornar la cantidad de pasos de distancia y el tamaño de cajas que acepta el casillero. No se deben realizar funciones adicionales recursivas o no recursivas, las mismas invalidan la solución (5.0 puntos).
c) Realizar la operación de unir la pila **u** (que se encuentra en el casillero) y la pila **p** (ingresada al almacén). Por tal motivo, solicite los códigos ordenados que se encuentran en la pila **u**. Recuerde que los productos más antiguos van en la parte más alta de la pila. El robot puede usar una pila auxiliar para mover las cajas, **pero no puede usar memoria adicional a la existente de ningún tipo,** sería como duplicar las cajas existentes. Para realizar esta operación el robot no necesita conocer el tamaño de las pilas. Para mostrar los resultados puede usar la función **imprimepila** (4.5 puntos).

Pregunta 3 (10 puntos)

Una empresa de venta de artículos por internet recibe pedidos de lunes a viernes y programa todas las entregas para los días sábados. Cada día, la empresa recibe un conjunto de pedidos, en donde los usuarios ingresan la hora en la que quisieran recibir sus pedidos el día sábado. Al término del día, la empresa ordena ascendentemente los pedidos realizados en ese día con respecto a la hora indicada. De esa forma, la empresa cuenta con una colección de pedidos ordenados por hora: una colección por cada día.
Por ejemplo, a continuación se muestra cómo estarían ordenados los pedidos al llegar al día sábado:

- Lunes: 8->10->12
- Martes: 9->11
- Miércoles: 8->9->10
- Jueves: 14->15->16
- Viernes: 17->18->19

Recuerde que los números indican la hora a la que el pedido es entregado el día sábado. Ese día, a primera hora, la empresa quiere organizar la entrega, por lo que requiere obtener una sola colección ordenada de todos los pedidos realizados en la semana. Para construir la colección ordenada debería tener en cuenta lo siguiente:

- Almacenar la información de los pedidos de cada día en una lista enlazada.
- Si dos pedidos tienen la misma hora de entrega, se debe colocar el pedido que fue hecho primero en la colección ordenada. Por lo tanto, un pedido que se hizo el día Lunes para ser entregado a las 8 debe ir antes que un pedido realizado el día Miércoles para ser entregado también a las 8.
- Cuando se tienen que fusionar dos colecciones con horarios disjuntos (el caso de jueves y viernes, en el ejemplo anterior), la fusión debería realizarse en tiempo constante, complejidad $O(1)$. Así también, en el peor de los casos, la fusión de dos listas debe tener complejidad $O(n)$.
- El algoritmo de fusión no puede emplear **memoria extra** (es decir, el algoritmo de fusión no puede usar malloc en ninguna parte), **tampoco estructuras adicionales, tipos abstractos de datos (TADs), arreglos o matrices**.

Consideraciones para la evaluación:

a) Definir la estructura de datos que soporte la aplicación e implementar el ingreso de los datos por teclado en las 5 listas que corresponden a los días de Lunes a Viernes. (2.0 puntos)
b) Implementar el algoritmo de fusión de listas. (7.0 puntos)
c) Implementar una función que muestre la lista final ordenada en donde, para cada elemento, se muestre la hora de entrega y el día en el que el pedido fue realizado (1.0 punto).

La salida para el ejemplo dado sería así:
8 (Lunes) -> 8 (Miercoles) -> 9 (Martes) -> 9 (Miercoles) -> 10 (Lunes) -> 10 (Miercoles) -> 11 (Martes) -> 12 (Lunes) -> 14 (Jueves) -> 15 (Jueves) -> 16 (Jueves) -> 17 (Viernes) -> 18 (Viernes) -> 19 (Viernes)

Profesores del curso: Iván Sipiran
Johan Baldeón
Rony Cueva

San Miguel, 6 de junio del 2020