PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA

Primer Examen

(Segundo Semestre 2021)

Duración: 2h 50 min.

- En cada función el alumno deberá incluir, a modo de comentario, la estrategia o forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- Los programas deben ser desarrollados en Ansi 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.
- Se utilizarán herramientas para la detección de plagios, por tal motivo si se encuentran soluciones similares, se anulará le evaluación a todos los implicados y se procederá con las medidas disciplinarias dispuestas por la FCI.
- Para este examen solo se permite el uso de las librerías **stdio.h, stdlib.h** y **math.h**
- **No están permitidas las funciones para obtener el tamaño de la pila o cola que recorran estas estructuras.**
- Su trabajo deberá ser subido a PAIDEIA.
- Los archivos deben llevar como nombre su código de la siguiente forma codigo_EX1_PY

Pregunta 1 (10 puntos)

Se desea implementar un programa que optimice las traslaciones del ascensor de un edificio de N pisos. Para ello, el programa utiliza una cola de solicitudes en el que cada solicitud tiene el piso al que debe dirigirse y su dirección si es de subida o de bajada. Luego que el ascensor llega a un piso puede haber más solicitudes para el ascensor, sea por las llamadas desde otros pisos o por las personas que ingresan al ascensor. Recuerde que cuando una persona está fuera del ascensor en un piso y desea solicitar al ascensor, puede presionar uno de los dos botones si es para subir o bajar, e incluso puede darse el caso de haber dos solicitudes desde un piso, sea para subir y bajar, o bajar y subir, pero una ocurre después de la otra. Si una persona presiona más una vez el mismo botón, solo se considera una solicitud y es encolada. Cuando una persona ingresa al ascensor y presiona uno de los pisos al que quiere dirigirse (que puede ser de subida o de bajada), esta solicitud es encolada.

Para que la optimización de las traslaciones del ascensor se realice, luego que se ingresan las solicitudes a la cola, estas deben de ordenarse de menor a mayor las solicitudes de subida superiores al piso donde se encuentra el ascensor hasta el máximo piso ingresado en la cola comenzando por la cabeza de la cola si es que la solicitud que está en la cabeza va de subida, y las solicitudes de bajada se ordenan de mayor a menor luego de las de subida. Si la solicitud que está en la cabeza de la cola va de bajada, entonces las solicitudes de bajada inferiores al piso donde se encuentra el ascensor se ordenan de mayor a menor hasta el mínimo piso ingresado hasta ese momento, y las solicitudes de subida se ordenan de menor a mayor luego de las de bajada.

Su programa realizará la simulación de las solicitudes y movimientos del ascensor mientras existan solicitudes en la cola. Asuma que al inicio el ascensor está en el primer piso y tiene en la

cola una solicitud ingresada que va de subida hacia el primer piso. Para la simulación, cuando el ascensor está en un piso, debe de indicar la cantidad de solicitudes a ingresar a la cola, por cada solicitud se indica el piso a dirigirse y el valor de la dirección (1 si es de subida o 0 si es de bajada).

A continuación, les mostramos un ejemplo del ingreso de datos que requiere este programa:

```
Ingrese la cantidad de pisos del edificio: 7
Movimientos del ascensor en un edificio de 7 pisos:

La cola de solicitudes del ascensor es:
1-1 ->
El ascensor está en el piso 1
Ingrese la cantidad de solicitudes: 4
Solicitud 1 [número_piso y dirección(1:sube,0:baja)]: 3 1
Solicitud 2 [número_piso y dirección(1:sube,0:baja)]: 6 1
Solicitud 3 [número_piso y dirección(1:sube,0:baja)]: 2 1
Solicitud 4 [número_piso y dirección(1:sube,0:baja)]: 4 1

La cola de solicitudes del ascensor es:
6-1 -> 4-1 -> 3-1 -> 2-1 ->
El ascensor está en el piso 2

Ingrese la cantidad de solicitudes: 2
Solicitud 1 [número_piso y dirección(1:sube,0:baja)]: 5 1
Solicitud 2 [número_piso y dirección(1:sube,0:baja)]: 1 0

La cola de solicitudes del ascensor es:
1-0 -> 6-1 -> 5-1 -> 4-1 -> 3-1 ->
El ascensor está en el piso 3

Ingrese la cantidad de solicitudes: 2
Solicitud 1 [número_piso y dirección(1:sube,0:baja)]: 1 0
Solicitud 2 [número_piso y dirección(1:sube,0:baja)]: 2 0

La cola de solicitudes del ascensor es:
1-0 -> 2-0 -> 6-1 -> 5-1 -> 4-1 ->
El ascensor está en el piso 4

Ingrese la cantidad de solicitudes: 1
Solicitud 1 [número piso y dirección(1:sube,0:baja)]: 3 0

La cola de solicitudes del ascensor es:
1-0 -> 2-0 -> 3-0 -> 6-1 -> 5-1 ->
El ascensor está en el piso 5

Ingrese la cantidad de solicitudes: 1
Solicitud 1 [número piso y dirección(1:sube,0:baja)]: 7 1

La cola de solicitudes del ascensor es:
7-1 -> 1-0 -> 2-0 -> 3-0 -> 6-1 ->
El ascensor está en el piso 6

Ingrese la cantidad de solicitudes: 1
Solicitud 1 [número piso y dirección(1:sube,0:baja)]: 4 0

La cola de solicitudes del ascensor es:
7-1 -> 1-0 -> 2-0 -> 3-0 -> 4-0 ->
El ascensor está en el piso 4
```

```
Ingrese la cantidad de solicitudes: 0

La cola de solicitudes del ascensor es:
7-1 -> 1-0 -> 2-0 -> 3-0 ->
El ascensor está en el piso 3
Ingrese la cantidad de solicitudes: 0

La cola de solicitudes del ascensor es:
7-1 -> 1-0 -> 2-0 ->
El ascensor está en el piso 2

Ingrese la cantidad de solicitudes: 0

La cola de solicitudes del ascensor es:
7-1 -> 1-0 ->
El ascensor está en el piso 1

Ingrese la cantidad de solicitudes: 0

La cola de solicitudes del ascensor es:
7-1 ->
El ascensor está en el piso 7

Ingrese la cantidad de solicitudes: 0

La cola de solicitudes del ascensor es:
No hay solicitudes en la cola del ascensor.
El ascensor está en el piso 7
```

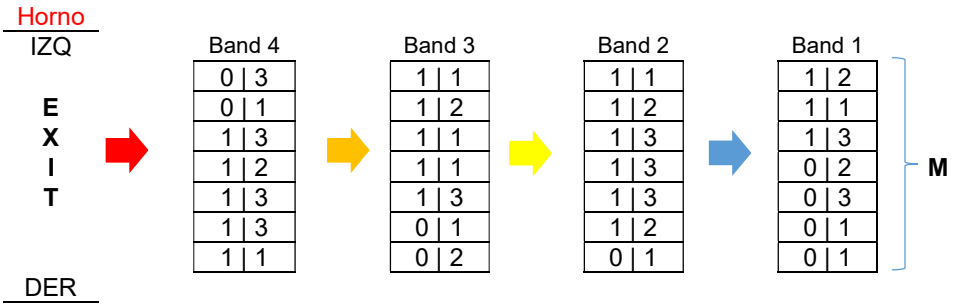
Se le solicita implementar lo siguiente:

a) Las estructuras de datos que permitan soportar lo descrito (1.0 puntos).
b) Una función que permita **ordenar** la cola de acuerdo con las consideraciones descritas. Para el ordenamiento de la cola sólo puede usar las operaciones de cola (encolar, desencolar, está vacía y visualizar el primero). **No pueden usar otras estructuras como arreglos, listas, pilas u otras colas para resolver el ordenamiento de la cola**, solo la misma cola de solicitudes. Puede usar iteraciones o recursividad (5.5 puntos).
c) Las funciones necesarias incluyendo el principal para realizar la simulación descrita (3.5 puntos).

Pregunta 2 (10 puntos)

La empresa “Pequeño Cesar” dedicada a la elaboración de comida ha desarrollado una nueva línea de cocción de pizzas. Esta línea se ha implementado utilizando 4 grandes bandejas que ingresan al horno al mismo tiempo, cada bandeja lleva **M** pizzas de diferentes tipos como son las pizzas Americana:1, Pepperoni: 2 y Hawaiana: 3.

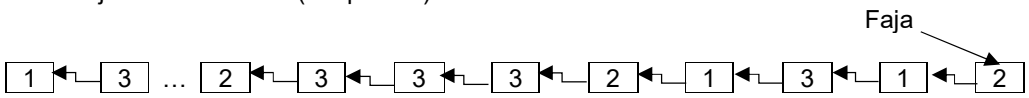
Debido a las diferentes temperaturas que se manejan dentro del horno muchas veces el calor puede ser más alto en cierta parte (izquierda o derecha), lo que puede originar que algunas pizzas que se encuentran a uno de los lados salgan un poco quemadas (se marcaran con 0, las buenas con 1), este problema solo puede ocurrir en uno de los lados para cada bandeja. A continuación, se aprecia un ejemplo de la línea al salir de horno, con un M = 7:



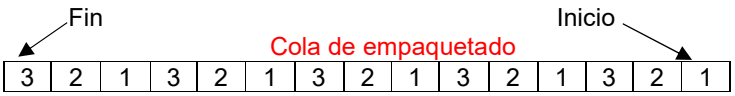
a) Utilizando la estrategia de **divide y vencerás,** calcule la cantidad de elementos **buenos** que tiene cada una de las bandejas luego de salir del horno, recuerden que siempre los productos quemados están a **un solo** lado de la bandeja. Para esta operación la complejidad debe ser de *O*(log M). Puede considerar cada bandeja como un arreglo ya que las posiciones son fijas (4.0 puntos)

Luego de esta revisión se pasa a un proceso de **clasificación,** el cual consiste en colocar los productos **buenos** de cada bandeja en una sección de una faja transportadora (las secciones de la faja funcionan como listas) de tal forma que al final se tienen 4 secciones de la faja separadas.

b) Desarrolle una función que se encargue de **cargar** los productos en cada sección de la faja (muestre este resultado) y finalmente desarrolle una función que se encargue de **unir** las secciones de la faja en una sola faja, esta última función debe tener una complejidad *O*(1), para esta operación no puede usar memoria adicional y solo puede usar las listas creadas. La faja resultante debe ser similar a la que se muestra a continuación, donde las pizzas de la bandeja 1 van adelante (3.0 puntos)



Una vez que los productos están en la faja pasan a un proceso de **selección**, para poder armar los combos que se ofrecen esta temporada. Esta temporada cada combo o mix se trata de los 3 tipos de pizza por el precio de una sola, y se venden dentro de una caja con 3 compartimentos. Por tal motivo los productos deben pasar a una cola de empaquetado formando grupos de 3, primero siempre una Americana, luego una Pepperoni y al final una Hawaiana, y así sucesivamente como se muestra la figura siguiente:



Si existen mas productos de un tipo, que no puedan formar un combo, se quedaran en los espacios de apilamiento provisional. Se sabe que para el proceso de selección se cuenta con 3 espacios de apilamiento como máximo.

c) Desarrollar el proceso de **selección** que genere como resultado la cola de empaquetado, a partir de la faja transportadora (lista) y la cantidad de combos completos que se pueden obtener con los datos suministrados. Para realizar este proceso puede emplear 3 pilas auxiliares como máximo que servirán como puntos de apilamiento provisional. Al final las

pizzas que no pueden formar un combo se deben dejar en las pilas, pero cada espacio solo puede tener un tipo de pizza (3.0 puntos)

Para desarrollar esta pregunta recuerde que se trata de simular objetos físicos, por tal motivo no duplique datos. Finalmente, no es necesario desarrollar un ingreso de información, puede usar los datos que se proporciona como ejemplo directamente.

Profesores del curso:

Johan Baldeon
Rony Cueva

San Miguel, 23 de octubre del 2021