UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTRUCTURAS DE DATOS Y ALGORITMOS I

M. en I. Fco. Javier Rodríguez García

Grupo

Parcial 4: Algoritmos y estructuras de datos

Alumno:	
Número de cuenta:	
Fecha:	
Calificación:	

Rev. 1.1

Esta carátula deberá ser llenada, impresa y entregada **con cada examen**

1 Objetivos

Que el alumno analice y sintetice problemas de la vida real relacionados con el uso de las estructuras de datos y algoritmos.

Conocimientos previos:

- Lenguaje C
- Tipos de datos definidos por el usuario
- Programación por capas
- Estructuras de datos
- Algoritmos

Nota:
Este examen no pone a prueba sus conocimientos, sino su capacidad de análisis y síntesis.

2 Ejercicios

2.1 Lista circular doblemente enlazada

Escribirá un simulador de un reproductor de música. Lo importante será moverse a través de su lista de reproducción hacia adelante y hacia atrás, ambas en forma circular, para mostrar información concerniente a cada canción.

2.1.1. Nodos

Cada nodo tendrá como información los siguientes campos (además, por supuesto, de los apuntadores hacia la derecha e izquierda):

- Índice: `int` (comienza en 1)
- Título de la canción: `char []`
- Álbum: `char []`
- Artista: `char []`
- Año: `int`

- Duración: `int` (en minutos)

2.1.2. Presentación de la información

Su aplicación mostrará un menú con las siguientes opciones:

1. **Salir**. Termina la aplicación. Cuando la aplicación termine toda la memoria dinámica deberá ser devuelta al sistema y el archivo de canciones actualizado (si es que no lo hizo antes).

2. **Buscar**. Pide al usuario el nombre de la canción y la busca en toda la lista. Si encuentra una coincidencia (o la primer coincidencia, en caso de existir varias canciones con el mismo nombre), entonces presenta la información completa (ver 2.1.1). En caso contrario le hace saber al usuario que dicha canción no está en la lista.

3. **Reproducir**. Muestra los datos de la canción a la que se quedó apuntando luego de una operación Adelante o Atrás.

4. **Adelante**. Se mueve a la siguiente canción y muestra la información completa (ver 2.1.1). Si llega al final se regresa al principio de la lista.

5. **Atrás**. Se mueve a la canción anterior y muestra la información completa (ver 2.1.1). Si llega al principio se regresa al final de la lista.

6. **Agregar**. Agrega una nueva canción al final de la lista y en el archivo de canciones.

7. (*Opcional, si tiene tiempo y ganas*) **Borrar**. Borra una canción (de la lista y del archivo de canciones) que coincida con el índice devuelto en la opción 2.

2.1.3. Llenado de la lista de reproducción

La carga inicial de las canciones debe hacerse desde un archivo. Éste puede ser de texto o binario, como Ud. lo decida.

2.1.4. Función Get()

Si lo considera necesario agregue una función **Get()** que devuelva los

datos de la canción actual (a la que quedó apuntando después de las operaciones Adelante y Atrás).

2.1.5. Funciones Next() y Prev()

Si lo considera necesario agregue un par de funciones **Next()** y **Prev()** que se muevan hacia adelante y hacia atrás en la lista enlazada.

2.1.6. Driver program

Su driver program llenará la lista de reproducción y posteriormente mostrará el menú de 2.1.1.

Entregables
Nombre a su módulo de lista doblemente enlazada circular como **lldblc.c** y **lldblc.h.** En papel entregará **lldblc.c**. Nombre a su driver program como **reproductor.c** y entréguelo en papel.

2.2 El problema del “cambio” y simulador de cajero automático

Escribirá una simulador de cajero automático enfocándose en la aplicación de un **algoritmo voraz** para calcular la cantidad de billetes que se entregará al tarjeta-habiente en cada transacción.

2.2.1. Monedas limitadas

El ejemplo clásico asume una cantidad ilimitada de monedas por lo que nunca se preocupa porque éstas se agoten (sin embargo sí se considera una cantidad finita para las denominaciones). En nuestro ejercicio la cantidad de monedas (o mejor dicho de billetes) sí será limitada. Para ver esto en un escenario real piense en el cajero automático. Éste se llena de vez en cuando con una cantidad limitada de efectivo en diferentes denominaciones. Cada vez que se realiza una transacción exitosamente la cantidad de billetes de cada denominación debe ser debitada. Cuando el cajero se queda sin efectivo avisa de esto y prohíbe cualquier transacción de retiro de efectivo. Si la operación no pudo completarse (porque así podría suceder si se termina alguna denominación) debera indicárselo al cliente de su código. Ud. decide el mecanismo para avisar que la operación se llevó a cabo o no.

Lista de denominaciones

La lista de denominaciones deberá ser una lista cuyos nodos tendrán como información los siguientes campos (además, por supuesto, el o los apuntadores al siguiente o siguientes nodos):

- Denominación
- Cantidad de billetes

Y como estructura de datos subyacente una lista enlazada.

Llenado inicial de la lista de denominaciones

Cuando su programa arranque deberá simular que llena el cajero con efectivo. Escribirá una función que pregunte al usuario (en este caso al empleado del banco) por la cantidad de billetes inicial para cada denominación. Las denominaciones permitidas son: 500, 200, 100, 50 y 20. Para ganar tiempo puede escribir una lista fija (arreglo estático) con

las denominaciones y cantidad de efectivo inicial, y posteriormente llenar la lista de denominaciones con la funciones de la API de la lista enlazada.

2.2.2. Conjunto solución

El conjunto solución será una cola con una lista enlazada simple como estructura de datos subyacente.

Si la transacción no puede ser completada debido a que no existe una solución posible, entonces se le avisará al usuario y el estado del cajero deberá ser idéntico al que tenía antes de comenzar la transacción.

Si la transacción tuvo éxito, entonces se actualizará la cantidad de billetes para cada denominación utilizada, así como el efectivo total remanente. Al usuario le será presentada la cantidad debitada y una impresión del conjunto solución.

2.2.3. Driver program

Su driver program inicialmente llenará el cajero automático según , y posteriormente presentará un menú con las siguientes opciones:

1. **Salir**. Sale de la aplicación.
2. **Debitar**. Pregunta al usuario por la cantidad a debitar y aplica 2.2.2.
3. **Ver efectivo remanente**. Muestra la cantidad total de efectivo remanente, no los billetes que aún quedan por cada denominación.

Entregables
Nombre a su módulo de cola como **queue.c** y **queue.h**, entregue este último en papel. Nombre a su módulo de lista enlazada como **ll.c** y **ll.h**. Nombre a su driver program como **cajero.c** y entréguelo. Entregue en papel la porción de código donde se encuentre su algoritmo voraz. Suponiendo que haya sido una función **Calcular(…)**, entonces deberá entregar el código de esta función. Si la llamó

diferente no hay ningún problema, siempre y cuando haga lo que se espera.

3 Referencias y bibliografía recomendada

- [BRASSARD97]
- [WIRTH85]
- [GALVE93]
- [BAASE02]
- [WIRTH76]
- [LÓPEZ09]
- Cualquier texto sobre estructuras de datos (no importa el lenguaje de programación del que hable).
- [JOYANES05]
- [DEITELxx]
  Deitel, H. M., Deitel, P. J. **Cómo programar en C/C++**. 2da. ed. ESPAÑA: McGraw-Hill, 20xx.
- Referencia del lenguage C
  http://www.cplusplus.com/reference/