ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
ESTRUCTURAS DE DATOS
PRIMERA EVALUACIÓN - II TÉRMINO 2017

Nombre: _____ ***Matrícula:*** _____

TEMA 1. (5 PUNTOS)

Describa en palabras el funcionamiento de las siguientes líneas de código y muestre su salida en pantalla:

```
Queue<Integer> q = new LinkedList<>();
q.offer(0);
q.offer(1);
for (int i = 0; i < 5; i++) {
        int a = q.poll();
        int b = q.poll();
        q.offer(b);
        q.offer(a + b);
        System.out.println(a);
}
```

TEMA 2. (10 PUNTOS)

Para cada uno de los siguientes escenarios escoja la "mejor" estructura de datos. Puede seleccionar una o una combinación de las siguientes y justifique su respuesta:

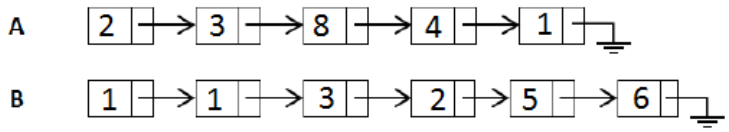
- Lista basada en arreglos
- Lista Simplemente Enlazada
- Lista Doblemente Enlazada
- Lista Circular Simplemente Enlazada
- Lista Circular Doblemente Enlazada
- Pila
- Cola
- Cola de Prioridad
- HashMap
- HashTable

1) Cabify lleva un registro de las carreras que han ocurrido en el día, la carrera tiene un número que se genera secuencialmente, además por cada carrera se almacena los datos del destino, origen, cliente, etc. Cabify en su sistema desea consultar los datos asociados a la carrera por el número de la carrera.
2) El rally Dakar es una competición mundial de rally, donde se toma el tiempo de salida del vehículo y el tiempo de llegada a la meta. Se desea premiar a los tres primeros vehículos que han obtenido los menores tiempos en llegar a la meta.
3) Facebook ha implementado la funcionalidad de "Personas que quizás conozcas", en esta sección se muestran todas las sugerencias de amigos con sus fotos de perfil y se puede ir revisando uno a uno los perfiles que sugiere facebook.
4) Los exploradores requieren un sistema satelital que les permite guardar la geolocalización del último sitio seguro que han visitado, en caso de que ocurra una emergencia, ellos quieren regresar a su punto de partida pasando por todos los sitios seguros para tomar las provisiones necesarias.
5) La lotería nacional ha decidido vender boletos electrónicos, los números de la lotería están conformado por 5 dígitos aleatorios, desde 12345 hasta 98765. La compra del boleto se realizará en línea solicitando los datos de facturación del comprador y el sorteo se realizará generando un número aleatorio para premiar al ganador.

TEMA 3. (15 PUNTOS)

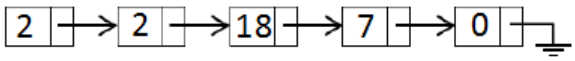
Implemente el método sumaMenores(LinkedList <Integer> A, LinkedList <Integer> B) que recibe dos listas A y B, y retorna una lista con la misma cantidad de elementos que A, en donde cada nodo tiene la suma de los primeros nodos de B cuyo valor es menor al valor del nodo correspondiente en A.

Por ejemplo: Sean las listas A y B



Valores de lista de retorno:
nodo 1: primeros valores menores a 2: 1 + 1 = 2
nodo 2: primeros valores menores a 3: 1 + 1 = 2
nodo 3: primeros valores menores a 8: 1 + 1 + 3 + 2 + 5 + 6 = 18
nodo 4: primeros valores menores a 4: 1 + 1 + 3 + 2 = 7
nodo 5: primeros valores menores a 1: = 0

Lista resultado:



TEMA 4. (20 PUNTOS)

Considere la siguiente definición de la Lista Doblemente Enlazada:

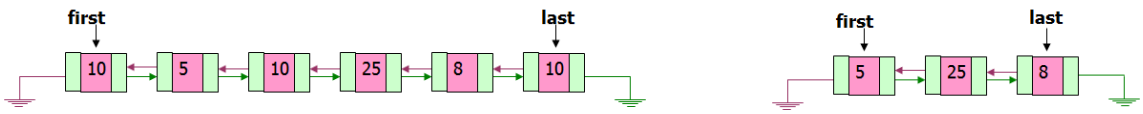
```
public class ListaDoblementeEnlazada<E> {
private NodoLista<E> first;
private NodoLista<E> last;
...
}
```

```
public class NodoLista<E> {
private E data;
private NodoLista<E> next;
private NodoLista<E> previous;
...
}
```

Se le solicita implementar en la clase ListaDoblementeEnlazada el siguiente método de manera **iterativa y recursiva**:

removeAll, que recibe un elemento y procede a eliminar todas las ocurrencias del elemento en la lista con un tiempo máximo de ejecución O(n/2).

```
public removeAll(E elemento)
```




```
l.removeAll(10)
```

TEMA 5. (25 PUNTOS)

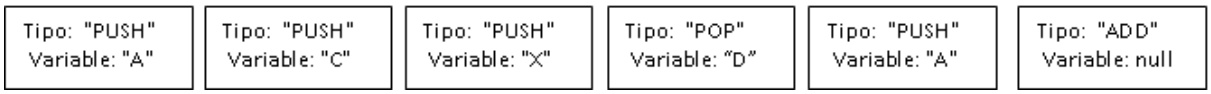
La arquitectura de pila es un tipo de arquitectura del procesador que no almacena operandos dentro del CPU, sino en una pila dentro de memoria. En esta arquitectura, existen dos instrucciones para operadores de pila: PUSH y POP. Además, las operaciones aritméticas (ADD, SUB, MUL, DIV) se realizan con los 2 elementos en el tope de la pila.

Asumiendo que se tiene la clase ***Instrucción*** que contiene los atributos ***tipo*** y ***variable***, donde ***tipo*** indica el nombre de instrucción a realizar ("PUSH", "POP", "ADD", "SUB", "MUL", "DIV") y ***variable*** representa el nombre de la variable con la que se requiere realizar una operación. Se dispone de un mapa ***variables***, que indica la dirección de memoria que representa cada variable. Implemente el método ***maquinaDePila*** que procesa una cola de instrucciones, además, recibe el mapa de variables y un arraylist con las direcciones de memoria (índice del arraylist) y sus valores (datos del arraylist).

```
public void maquinaDePila(Queue<Instruccion> cola, Map<String,Integer> variables, ArrayList<Integer> memoria)
```

Por ejemplo:

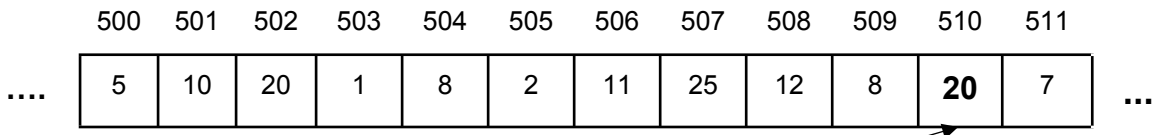
cola:

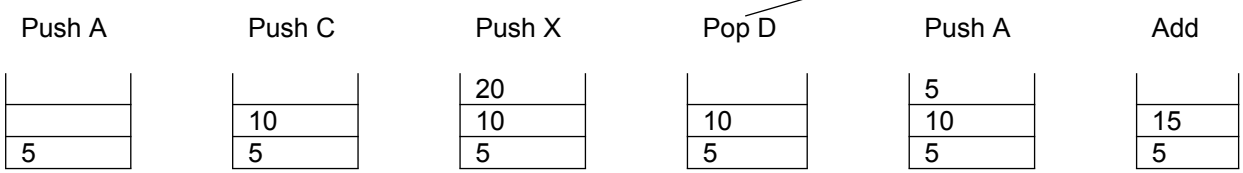


variables:

{"A" = 500, "B"= 504, "X"= 502, "D"= 510, "C"= 501}

memoria:





TEMA 6. (25 PUNTOS)

Mediante el siguiente TDA se han representado los procesos que ejecuta un procesador en un computador:

```
public class Proceso {
private int id; // identificador del proceso
private String nombre; // nombre del proceso
private int ciclos; // número de ciclos que requiere del procesador
private int prioridad; //prioridad del proceso
...
}
```

Usted deberá simular el comportamiento de un procesador que atiende los procesos acorde a su prioridad (aquellos procesos que tengan una mayor prioridad serán los primeros en atenderse). El procesador solo puede atender un ciclo del proceso a la vez; sí el proceso no tiene ciclos, se entiende que el proceso ha terminado. Los procesos pueden ser interrupciones (nombre de proceso que empieza con interrupción), cuando el procesador atiende una interrupción deberá ejecutarse la pila de procesos de esa interrupción que se encuentran en su handler (manejador de interrupciones). Los procesos que están en la pila de procesos de la interrupción no tienen prioridad, por lo tanto, serán atendidos por el procesador conforme a un comportamiento LIFO.

Nota.- Los procesos que son interrupciones siempre requerirán un ciclo del procesador.

```
public void simulacion(List<Proceso> procesos, Map<Integer,Stack<Proceso>> handler)
```

Ejemplo:

```
procesos = [101-chrome.exe-2-10, 302-interrupción1.exe-1-10, 403-firefox.exe-2-8, 503-
interrupción2.exe-1-6, ...]
handler = {302:[602-proceso2.exe-2,601-proceso1.exe-3], 503:[604-proceso3.exe-2,604-
proceso4.exe-1]}
```

```
Procesando: Id: 101 - Nombre: chrome.exe - Ciclos Actuales: 10 - Ciclos restantes: 9
Procesando: Id: 302 - Nombre: interrupción1.exe - Ciclos Actuales: 1 - Ciclos restantes: 0
Procesando: Id: 601 - Nombre: proceso1.exe - Ciclos Actuales: 3 - Ciclos restantes: 2
Procesando: Id: 601 - Nombre: proceso1.exe - Ciclos Actuales: 2 - Ciclos restantes: 1
Procesando: Id: 601 - Nombre: proceso1.exe - Ciclos Actuales: 1 - Ciclos restantes: 0
Procesando: Id: 602 - Nombre: proceso2.exe - Ciclos Actuales: 2 - Ciclos restantes: 1
Procesando: Id: 602 - Nombre: proceso2.exe - Ciclos Actuales: 1 - Ciclos restantes: 0
Procesando: Id: 101 - Nombre: chrome.exe - Ciclos Actuales: 9 - Ciclos restantes: 8
Procesando: Id: 101 - Nombre: chrome.exe - Ciclos Actuales: 8 - Ciclos restantes: 7
……..
```