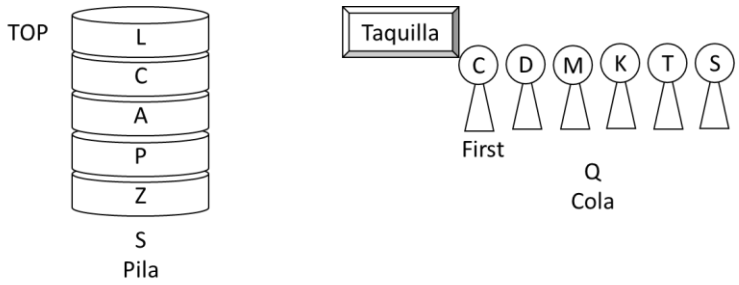
Estructura de Datos 2024-1S
Departamento de la Ciencias de la Computación y la Información
Examen 3 – 27 de agosto 2024

Nombre: _____

**Instrucciones:** lea con atención las siguientes instrucciones. En los puntos que solicitan seudocódigo, debe presentar la representación de alto nivel del paso a paso lógico que se requiere para la solución del problema. Por tanto, no requiere usar un lenguaje de programación especifico; sin embargo, puede usar palabras reservadas de los lenguajes de programación, tales como IF, ELSE, FOR, WHILE, etc. Para la solución del examen puede asumir las implementaciones de las clases FECHA, DIRECCION, USUARIO, NODE, LIST, DOUBLENODE, DOUBLELIST, QUEUE, STACK.

**Parte 1: Pilas y Colas (30 puntos).** Para las siguientes preguntas asuma la pila S y la cola Q presentada en la figura. Asuma que tanto la pila y la cola se encuentran implementadas con base a una lista simple, y por tanto hacen uso de memoria dinámica. Indique su respuesta dentro del cuadro de cada pregunta (5 puntos cada una).




1. Partiendo de la pila S de la figura, el tope de la pila después de aplicarle las siguientes operaciones es: **C**

```
1. S.push('D')
2. S.pop()
3. S.pop()
```

2. Partiendo de la cola Q de la figura, el primer elemento de la colección después de realizar las siguientes operaciones es: **D**

```
1. Q.dequeue()
2. Q.first()
3. Q.enqueue('Z')
```

3. Partiendo de la pila S y cola Q de la figura, la operación de la línea 5 retorna: **M**

```
1. Q.dequeue()
2. Q.enqueue(S.pop())
3. Q.dequeue()
4. S.push(Q.dequeue())
5. S.top()
```

4. Partiendo de la pila S y cola Q de la figura, la operación de la línea 5 retorna: **6**

```
1. Q.enqueue(S.pop())
2. Q.dequeue()
3. Q.enqueue(S.top())
4. S.push(Q.dequeue())
5. Q.size()
```

5. Partiendo de la pila S y la cola Q de la figura, la operación de la línea 4 retorna: **0**

```
1. while(!Q.isEmpty())
2.     S.push(Q.dequeue())
3. while(!S.isEmpty())
4.     S.pop()
5. S.size()
```

6. Partiendo de la pila S y la cola Q de la figura, la operación de la línea 4 retorna: **Null**

```
1. while(!S.isEmpty() && !Q.isEmpty())
2.    S.pop()
3.    Q.dequeue()
4. S.top()
```

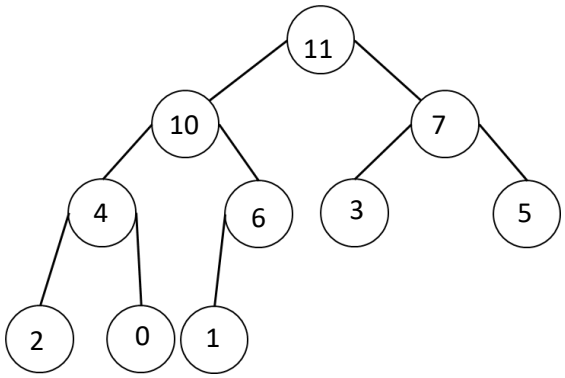
Estructura de Datos 2024-1S
Departamento de la Ciencias de la Computación y la Información
Examen 3 – 27 de agosto 2024

Nombre: _____

**Parte 2: Heap (30 puntos).** Para las siguientes preguntas considere el arreglo A presentado a continuación:

| 6 | 1 | 5 | 4 | 11 | 3 | 7 | 2 | 0 | 10 |
|---|---|---|---|---|---|---|---|---|---|

1. Complete el siguiente grafico con el max-heap que se obtiene al aplicar el método BUILD-MAX-HEAP al arreglo A (10 puntos):

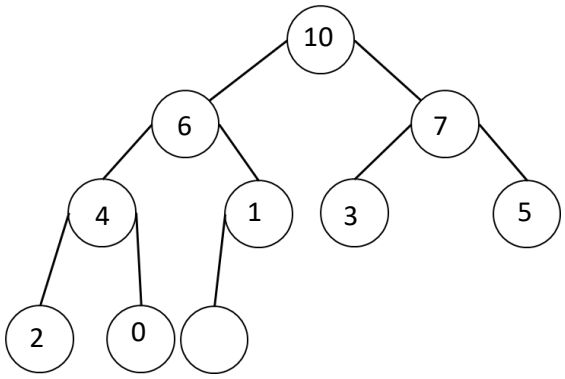



2. El algoritmo HEAPSORT() presentado en el siguiente seudocódigo, realiza construye un max-heap como el obtenido en el punto punto 1. A partir de este resultado, presente como queda el arreglo A después de realizar dos iteraciones del ciclo FOR (líneas 3-8) (10 puntos):

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

| 7 | 6 | 5 | 4 | 1 | 3 | 0 | 2 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|

3. Complete el siguiente grafico con el max-heap que se obtiene al aplicar el método HEAP-EXTRACT-MAX al heap obtenido en el punto 1 (10 puntos):



Estructura de Datos 2024-1S
Departamento de la Ciencias de la Computación y la Información
Examen 3 – 27 de agosto 2024

Nombre: _____

**Parte 3. Aplicaciones de estructura de datos (15 puntos):** El seudocódigo presentado a continuación determina si una expresión de paréntesis (cadena con paréntesis de apertura y cierre) esta balanceada empleando una pila S. **Presente la modificación del algoritmo** para verificar si una expresión con paréntesis “( )”, llaves “{ }” y corchetes “[ ]” esta balanceada. Por ejemplo:

- P = [[()]{()()}] Expresión balanceada
- P = [{()([])}] Expresión no balanceada

```
Balanceo(String P)
1.  Stack S = new Stack()
2.  for (int i=0; i<P.length; i++)
3.     if P.chatAt(i)=='('
4.        S.push('(')
5.     else
6.        if S.isEmpty()
7.           return false
8.        else
9.           S.pop()
10. if S.isEmpty()
11.    return true
12. else
13.    return false
```

Balanceo(String P)

1. Stack S = new Stack()
2. for (int i=0, i<P.length, i++)
3. if P.charAt(i)==’(’ || if P.charAt(i)==’[’ || if P.charAt(i)==’{’
4. S.push(P.charAt(i))
5. elseif (S.isEmpty())
6. return FALSE
7. elseif P.charAt(i)==’)’ && S.top()==’(’
8. S.pop()
9. elseif P.charAt(i)==’]’ && S.top()==’[’
10. S.pop()
11. elseif P.charAt(i)==’}’ && S.top()==’{’
12. S.pop()
13. else
14. return FALSE
15. if S.isEmpty()
16. return true
17. else
18. Return FALSE

Estructura de Datos 2024-1S
Departamento de la Ciencias de la Computación y la Información
Examen 3 – 27 de agosto 2024

Nombre: _____

**Parte 4. Aplicación de colas (15 puntos**). Un banco requiere un sistema para asignación de turnos. Para el diseño del sistema, el banco solicita que cuando hayan 10 o menos personas esperando por atención en la sucursal, estas sean atendidas por orden de llega, es decir, usando el principio FIFO. Pero cuando dentro del banco hayan más de 10 clientes, la atención se debe realizar por prioridad, donde la prioridad está dada por la edad del cliente y el numero de años con productos en el banco: p = edad + años_como _cliente. Para solucionar este problema se diseñan las clases Cliente y TurnoUsuario presentadas en la figura. La clase Cliente mantiene el nombre, número de identificación y prioridad p de cada usuario que llega al banco a pedir turno. La clase TurnoUsuario mantiene una cola Q, la cual mantendrá clientes mientras solo haya 10 o menos usuarios en el banco, y un heap H que manejará la prioridad cuando hayan más de 10 clientes en el banco. La figura presenta las clases Max-Heap, Cliente, y TurnoUsuario.

Clase Max-Heap

| Max-Heap |
| --- |
| - A: Object[ ]<br>- Size: int |
| + isEmpty(): Boolean<br>+ size(): int<br>+ Max-Heapify(int i)<br>+ Max-Heap-Insert(Object c)<br>+ Heap-Extract-Max(): Object<br>+ Heap-Maximum(): Object |

Clase Cliente

| Cliente |
| --- |
| - nombre: String<br>- id: long<br>- p: int |
| + setNombre(String n)<br>+ setId(long id)<br>+ setPrioridad(int p)<br>+ getNombre(): String<br>+ getId(): long<br>+ getPrioridad(): int |

Clase TurnoUsuario

| TurnoUsuario |
| --- |
| - Q: Queue<br>- H: Max-Heap<br>- num_turnos: int |
| + registrar(Cliente c)<br>+ atenderSiguiente(): Cliente |

a. Presente el seudocódigo para el método que permite registrar un cliente en el sistema de atención, el cual debe cumplir con las especificaciones del banco. Es decir, presente el seudocódigo para método registrar(Cliente c) de la clase TurnoUsuario.
b. Presente el seudocódigo para el método atenderSiguiente, el cual retorna que cliente que debe ser atendido según las reglas del banco.

```
Registrar(Cliente c)
1. num_turnos = num_turnos+1
2. if num_turnos <= 10
3.     Q.enqueue(c)
4. else
5.      while(!Q.isEmpty())
6.          H.Max-Heap-Insert(Q.dequeue())
7.     H.Max-Heap-Insert(c)
```

```
atenderSiguiente()
1.  if num_turnos<=10
2.     temp = Q.dequeue()
3.  else
4.     temp = H.Heap-Extract-Max()
5.  num_turnos = num_turnos – 1
6.  if num_turnos==10
7.     while(!H.isEmpty())
8.          Q.enqueue(H.Heap-Extract-Max())
9.  return temp
```

Estructura de Datos 2024-1S
Departamento de la Ciencias de la Computación y la Información
Examen 3 – 27 de agosto 2024

Nombre: _____

**Parte 5. Uso de estructura de datos (10 puntos):** Para cada una de las siguientes situaciones indique cuál de las estructuras de datos es las mas adecuada en términos de eficiencia computacional para solucionar el problema y administrar las colecciones requeridas. Seleccione entre arreglo de bajo nivel (tamaño fijo), lista doble, pila, cola, o heap.

| Descripción del problema | Estructura de datos |
|---|---|
| Se desea desarrollar un app de juegos educativos, la cual permita a los profesores realizar concursos durante las clases. Después de cada juego la app le reporta al profesor los 5 estudiantes con los mejores puntajes. | Arreglo de bajo nivel |
| Debido a los recursos limitados de una sala de urgencia, una clínica decide instalar un sistema en el cual a cada paciente que llega se le asigna un valor de prioridad de acuerdo al tipo de enfermedad y cuidados inmediatos que requiera. El sistema organiza la atención de los clientes de acuerdo a la prioridad asignada. | Heap |
| La cafetería de la universidad desea sistematizar los pedidos que realizan los estudiantes y profesores. A través de un chatbot, los estudiantes y profesores desde cualquier parte del campus pueden realizar su orden de almuerzo en la cafetería y este indica un tiempo estimado para que sea recogido. El sistema organiza los pedidos de acuerdo al orden de llegada. | Cola |
| Una cooperativa de ahorro solicita el desarrollo de un sistema que le permita a sus usuarios de forma fácil consultar los últimos 5 movimientos de su cuenta a través de un mensaje de texto. | Pila |
| Una empresa requiere un sistema para administrar los clientes mas frecuentes. Sin embargo, el número de clientes puede variar significativamente en el año. | Lista doble |