EXAMEN PARCIAL

Assignatura: **Estructura de dades**

Data: 4 d'abril de 2017 Curs: **2016/2017**

UNIVERSITAT DE BARCELONA

Nom: _____ **Cognoms**:_____ **NIUB/DNI**: _____

Aula: ____ **Fila**: ____ **Columna**: _____

Preguntes i Problemes **Temps: 90 minuts**

1. **(1 p) Partim d'una cua enllaçada (`LinkedQueue`) amb una representació amb enllaços simples (`SingleNode`). Aquesta `LinkedQueue` té un punter al primer element de la cua anomenat `head` i un punter al darrer element de la cua anomenat `rear`. Implementeu el mètode de la classe LinkedQueue anomenat `void concatena (LinkedQueue &a)`, que permet concatenar tots els elements de la `LinkedQueue a` al final de la cua. La `LinkedQueue a` no es modifica durant el procés de concatenació. L'especificació de la classe `SingleNode` és la següent:**

```
class SingleNode
{    private:
        char element;
        Node *next;
    public:
        Node(char e);
       ~Node();
        const char & getElement() const;
        Node * getNext()const;
        void setNext(Node * elem);
};
```

Es demana que la implementació tingui un cost O(1). Justifica la teva codificació amb dibuixos, pas a pas.

```
void LinkedQueue::concatena(LinkedQueue &a const)
{

if (a.empty()) throw new out_of_range("Emtpy queue");
 else
 {
      this->rear->setNext(a.head);
      this->rear = a.rear;
 }

}
```

2. **(0.5p) L'operació d'inserir un element en una llista enllaçada ordenada trigarà el mateix temps d'execució que l'operació d'inserir un element en un vector ordenat. Indica si l'afirmació és certa o falsa i justifica la teva resposta**

El cost computacional teòric és o(n) en els dos casos. En canvi, el temps d'execució que trigarà cadascuna de les operacions serà diferent.

Inserir en una llista enllaçada consisteix en buscar la posició i col·locar el nou element.
Inserir en un vector consisteix en buscar la posició, desplaçar els n elements fins al final i posteriorment col·locar el nou element a la posició que s'ha trobat.

Per tant, serà una mica més costós en temps d'execució (no en canvi en cost teòric computacional) inserir en un vector ordenat.

3. **(1 p) Defineix que és una cua prioritària. Indica quin és el contingut de la cua prioritària, com s'insereix els elements i com s'extrauen els elements de la cua. Finalment, posa un exemple d'ús de la cua prioritària.**

En una cua prioritària es guarden entrades (valor, clau)
Els elements s'insereixen en qualsevol ordre
Els elements s'extrauen segons el que tingui més prioritat (removeMin)

Un exemple d'ús seria ... per prioritzar els aterratges a l'aeroport. Un altre exemple seria ... per prioritzar els processos dels sistemes operatius.

4. **(2 p) Partim d'una STACK amb una representació amb enllaços simples que té un punter al top de la pila anomenat `front`. Implementeu el destructor de la pila utilitzant les funcions pròpies de la pila: size, empty, push, pop, top. A més a més, es demana que implementeu les operacions que heu cridat de la pila en la implementació del destructor de la pila. Assumiu que els nodes (`SingleNode`) estan definits tal i com s'especifica a l'exercici 1.**
   S'aconsella que useu dibuixos per justificar la vostra codificació. Quines excepcions heu llençat en la vostra codificació?

```
LinkedStack::~LinkedStack()
{
  while (!this->empty())
        pop();

}
// Aquí a sota les operacions que heu cridat en el destructor


bool LinkedStack::empty() const
{
    //if (!front) cout << "Empty LinkedStack " << endl;
    return (!this->front);  // this->front == nullptr
}

void LinkedStack::pop()
{
if   (this->empty())   throw   out_of_range("LinkedStack<>::pop():   empty
stack");
    else
    {
        Node<Element> * aux_node = front;
        front = aux_node->getNext();
        delete aux_node;
    }
    this->num_elements--;

}
```

5. (2p) Partint d'una ArrayStack<char> pila que guarda els caràcters d'una paraula. Escriu un mètode en C++ que indiqui si aquesta paraula és un palíndrom. L'especificació de la pila és la següent:

```
template <class E>
class ArrayStack {
public:
      ArrayStack();
      int size() const;
      bool empty() const;
      const E& top() const;
      void push(const E& e);
            void pop();
};
```

Noteu que la pila no disposa de constructor còpia. No es pot usar cap altra estructura de dades, només piles. La pila d'entrada ha de quedar intacta després de comprovar si el seu contingut és un palíndrom.

```
LA SOLUCIO NO ES UNICA. AQUI TENIU UNA POSSIBILITAT

bool esPalindrom(ArrayStack<char> &pila)
{
    ArrayStack<char> pila2 = ArrayStack<char>();
    ArrayStack<char> pila3 = ArrayStack<char>();
    ArrayStack<char> pila_aux = ArrayStack<char>();

    bool iguals = true;

        while (not pila.empty()){
            pila2.push(pila.top());
            pila3.push(pila.top());
            pila.pop();
        }

        while (!pila2.empty())
        {
            pila.push(pila2.top());
            pila_aux.push(pila2.top());
            pila2.pop();
        }

        while (!pila_aux.empty() and !pila3.empty() and iguals)
        {
            if (pila_aux.top() != pila3.top() ) iguals = false;
            else
            {
                pila_aux.pop();
                pila3.pop();
            }
        }
   return iguals;

}
```