

Enunciado:

Resuelva los siguientes ejercicios en C++14 sobre apuntadores, arreglos y clases. Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores.

1. Implemente una versión sencilla del contenedor secuencial `std::stack`, llamada `Stack`, usando el concepto de listas enlazadas (*linked lists*). Los métodos y variables de instancia que deben ser implementados se muestran a continuación.

```
1 | template <typename dataType>
2 | class Stack {
3 | private:
4 |     struct Cell {          /* Type for linked list cell */
5 |         dataType data;
6 |         Cell *link;
7 |     };
8 |
9 |     Cell *stack;          /* Beginning of the list */
10 |    int count;             /* Number of elements */
11 |
12 |    void deepCopy(const Stack<dataType> & src);
13 |
14 | public:
15 |     Stack();
16 |     ~Stack();
17 |     size_t size();
18 |     bool empty();
19 |     void clear();
20 |     void push(dataType & ch);
21 |     dataType pop();
22 |     dataType peek();
23 | };
```

Si considera que más métodos son esenciales en su implementación, no dude en incluirlos.

2. Implemente una versión simplificada del contenedor secuencial `std::queue`, llamada `Queue`, usando arreglos dinámicos (*dynamic arrays*). Los métodos y variables de instancia que deben ser implementados se muestran a continuación.

```
1 | template <typename dataType>
2 | class Queue {
3 | private:
4 |     static const int INITIAL_CAPACITY = 10;
5 |
6 |     dataType *array;
7 |     int capacity, head, tail;
8 |
9 |     void expandCapacity();
```

```
10     void deepCopy(const Queue<dataType> & src);  
11  
12 public:  
13     Queue();  
14     ~Queue();  
15     int size();  
16     bool empty();  
17     void clear();  
18     void enqueue(dataType & ch);  
19     dataType dequeue();  
20     dataType peek();  
21 };
```

Si considera que más métodos son esenciales en su implementación, no dude en incluirlos.