Name: Kormoua Khongmeng
Neptun Code: I3MLPQ

Basic of Programming 2

Laboratory report 5

Task 1.

1. The implementations in Set.cpp source file should be based on declarations (see the Set.h header file).

   Ans: Source codes and header file (with a very clear comments) are in the same folder.

   - A class Set has a following items as an (private) attribute:
     1. A number which will indicate how many element in an array of a set.
     2. An array of a set itself.

   - A class Set has a following items as a constructors and destructor:
     1. A initialize constructor which will make an instance of a set to be in the initial state.
     2. A copy constructor which will copy every information (2 attributes that we mentioned earlier) from a given instance of a class to a new instance of a class that we want to create.
     3. A destructor which will release all the allocated memory and delete every information of an instance of a class. (I also check if a given set has more than 1 element in the array, I need to use *delete*[] for this array. But if the given set has 1 or no element in the array then I simply use *delete* to release the allocated memory)

   - A class Set has a following items as a methods:
     1. An insert method, which will add a new given element to an instance of a class (if there is no such element inside a set yet), and also increase a number which indicate how many element in the array of that instance. (for this I first check if there is a given element in a set. If there is no, then I allocate a temporary memory space and copy the original array to a temporary one then add the new element at the end of the temporary array. After that delete the original array and make the temporary array as a original one).
     2. A remove method, which will remove a given element from the array of the set (if that element exist in the set) and decrease a number of element in the array of a set. (for this first I check if the given element

exist in the array of a set. If it is, then allocate a temporary memory space and copy every element from the original array to the temporary array except the given element that should be exclude from this array).

3. An isElement method, which will check if a given element exist in the array of a set. If not, return false. Otherwise, return true.
4. A print method which will print every element in the array of a Set to a screen.
5. An empty method, which will release the allocated memory of a set, an delete all information from the attribute as well. (for this I also check if a given set has more than 1 element in the array, I need to use *delete*[] for this array. But if the given set has 1 or no element in the array then I simply use *delete* to release the allocated memory)

2. Run the test cases inSetMain.cpp. Observe the program execution in debugger mode. Make sure that the execution result is correct.

   Ans: Everything works like expected.

3. Add an empty() method to the set class, which task is to clear the set (delete all elements). What is the difference between empty() function and a class destructor?

   Ans: A difference between empty() function and a class destructor are:

   - An empty() function needs a programmer to call for it, it is not called automatically like destructor. Therefore, if we want to delete an instance of a class during a program, we can use empty() function. However, we cannot do so with a destructor, since we cannot call a destructor by ourselves, it will be called only at the end of a scope that the instance of a class is declared.
   - A destructor cannot have any parameter and does not return anything. Unlike an empty() function that we could have parameter and can return something (if we want it to do so)

Homework.

Develop and implement a *FIFO* class. Use the *Set* class from Task 1 as an example. Consider the following mandatory functions and restrictions:

1. A *push* operation inserts a new element to FIFO, and a *pop* operation returns the firstly entered element.
2. Follow the principles of encapsulation: do not allow making the container inconsistent from the outside.
3. Dynamic memory management is to be used.
4. Based on the tests that can be found in SetMain.cpp, create test cases for FIFO class.

Ans: Source codes and header file (with a very clear comments) are in the same folder.

- A class FIFO has a following items as an (private) attribute:
    1. A number which will indicate how many element in an array of the FIFO.
    2. An array of a FIFO itself.

- A class FIFO has a following items as a constructors and destructor:
    1. A initialize constructor which will make an instance of a FIFO to be in the initial state.
    2. A copy constructor which will copy every information (2 attributes that we mentioned earlier) from a given instance of a class to a new instance of a class that we want to create.
    3. A destructor which will release all the allocated memory and delete every information of an instance of a class. (I also check if a given FIFO has more than 1 element in the array, I need to use *delete*[] for this array. But if the given FIFO has 1 or no element in the array then I simply use *delete* to release the allocated memory)

- A class FIFO has a following items as a methods:
    1. A push method, which will add a new given element to the end of an array of an instance of a class, and also increase a number which indicate how many element in the array of that instance. (for this I allocate a temporary memory space and copy the original array to a temporary one, then add the new element at the end of the temporary array. After that delete the original array and make the temporary array as an original one).
    2. A pop method, which will remove a first element from the array of the FIFO, decrease a number of element in the array of a FIFO and return the first element of the array. (for this, I check first whether the array of FIFO is empty or not, if not then, allocate a temporary memory space and copy every element from the original array to the temporary array except the first element that should be exclude from this array and finally return the value of the first element of the array).
    3. An isElement method, which will check if a given element exist in the array of a FIFIO. If not, return false. Otherwise, return true.
    4. An empty method, which will release the allocated memory of a set, an delete all information from the attribute as well. (for this I also check if a given FIFO has more than 1 element in the array, I need to use *delete*[] for this array. But if the given FIFO has 1 or no element in the array then I simply use *delete* to release the allocated memory)
    5. A print method, which will print every element of the array to the screen, if the FIFO is not empty. If the FIFO is empty then print that it is empty.