CS 200 – Introduction to Programming

Assignment 1 Part 2

Due Date: Saturday October 8 – 11:55 PM

Please keep in mind the following:

- Do not share your code with anyone else. All assignments are to be done individually.
- You must be able to explain any part of your submitted code.
- You must submit the **.cpp** file only. Make sure you name the files as:
  *<roll number>*_A1P*<part number>*.cpp
  - For example: 24100116_A1P2.cpp
- Make sure you follow the naming schemes of the .cpp files correctly. Failure to do so will result in getting a **0** in a question.
- No submissions other than via the LMS assignments tab will be entertained.
- Make sure your code compiles and runs. If a piece of code fails to compile, you'll be given a **0** in that question.
- All submissions will be checked for plagiarism. You are **not** allowed to copy code from the internet.

Question 1: Dynamic Arrays and Vectors – 100 Marks

In this question, you have to create a Vector class which mimics the functionality of the built-in vector class in C++. You have to complete the following tasks:

1. Create the following variables. Make sure that they are **private** else you'll not be awarded full marks in this section. **5 Marks**
    a. array: This will be an int* variable that will point to the start of the array.
    b. current_size: This will store the current number of elements that are present in the array.
    c. capacity: This will store the maximum capacity of the array.
    d. type: This will be a character variable which stores either 'q' or 's'. 's' means that the vector should behave like an array in the push() and pop() functions and 'q' means that it should behave like a queue.

You are required to create the following functions:

2. **Vector()** **7 Marks**
    Create a default constructor Vector() for this class. It must dynamically allocate an array of **SIZE=10** initially and set the values of other class variables appropriately. The default value for type must be 's'.
3. **~Vector()** **7 Marks**
    Create a destructor ~Vector() which frees up the memory allocated to the array.
4. **Vector(Vector &anotherVector)** **7 Marks**
    Create a copy constructor which makes a deep copy of another vector. When making a deep copy, you don't just copy the pointers; you must copy the actual values.
5. **int getLength()** **2 Marks**
    Make a function called int getLength() which returns the current numbers of elements in the vector.
6. **bool isEmpty()** **2 Marks**
    Make a function called isEmpty() which takes no arguments and returns a *bool*. It will return true if the array is empty else it'll return false.
7. **bool isFull()** **2 Marks**
    Make a function called isFull() which returns a *bool*. It will return true if the array is full else it'll return false.

8. **void increaseCap()** **17 Marks**

   Make a **private** function increaseCap() which doubles the size of the array and copies over the new elements. Make sure you update the respective variables.

9. **void changeType(char new_type)** **5 Marks**

   Make a function changeType(char new_type) which updates the type of the array and sets it equal to new_type.

10. **void insertAtHead(int num)** **3 Marks**

    Make a function insertAtHead(int num) which inserts a number at the start of the array. If the array is full, it should call increaseCap() first and then insert the element.

11. **void insertAtTail(int num)** **2 Marks**

    Make a function insertAtTail(int num) which inserts a number at the end of the array. If the array is full, it should call increaseCap() first and then insert the number. Inserting a number at the end of array would mean right after the last element of the array i.e. in simpler terms, the number added should be after 'current_size' instead of at 'capacity' of array.

12. **void deleteHead()** **10 Marks**

    Make a function called deleteHead() which deletes the first element in the array and adjusts the array appropriately.

13. **int getHead()** **2 Marks**

    Make a function called int getHead() which returns the first element in the array.

14. **int getTail()** **2 Marks**

    Make a function called int getTail() which returns the last element in the array.

15. **void deleteTail()** **2 Marks**

    Make a function called deleteTail() which deletes the last element in the array.

16. **void push(int num)** **10 Marks**

    Make a function called push(int num) which adds an element at the end of the array. You are also required to consider if the type is a queue or a stack and call the appropriate insert function.

17. **int pop()** **20 Marks**

    Make a function called int pop() which returns the first element in case of a queue and the last element in case of a stack. Make sure you also remove that element from the array.