CPSC 121 Fall 2021 Lab Final Exam

Instructor: Daniel Olivares

Instructions

- **You must work individually on this exam. This is a non-collaborative exam.**
- You will have the full 120 minutes finals period to complete the exam.
- You are free to use your laptop for this exam. **You will need to use the lab machine if your laptop is not functional.**
- This Exam is OPEN RESOURCE, i.e., you may use any resource (notes/course materials) **but NOT help from your neighbors/friends/etc.**
    - *I strongly encourage you to not get caught up in resource hunting during this time*
    - If you are caught collaborating with another student or outside individuals, **you will receive a 0 on the exam and the incident will be reported to the academic integrity board for disciplinary action. There will not be a warning or alternate consequences.**

Submitting the Exam

1. Intermittently paste your code into the "just in case" text submission box. This will "save" your code in case there is an issue with the file upload. Canvas "saves" the text in the textbox when you modify it. **This is only an emergency precaution. This does not mean you don't have to upload your files!**
    a. Canvas should alert you with an additional confirmation prompt if your file was not uploaded successfully.
    b. It is your responsibility to make sure you are using this text submission box so you do not lose your work if your computer crashes 100 minutes into the exam.
2. Make sure to compile and run your program one final time before submitting!
    a. Your project must build properly. **The most points the lab final can receive if it does not build properly is 65 out of 100.**
    b. If you are struggling to get your code to compile, I suggest selectively commenting out the code causing compile errors **and leave comments explaining why it is commented out**.
3. Use the appropriate file upload question to upload your **properly named** .cpp file.
    a. You should change lastname with your last name and remove the "Starter" from the filename, e.g., `lastname_LabFinalExamStarter.cpp` → `realLastName_LabFinalExam.cpp`

Grading Guidelines

The lab final is worth 100 points. Your submission will be evaluated based on a successful compilation and adherence to the program requirements.

Your code will be tested in 2 parts: 1) I will compile your program file and compare it **with my solution and grade based on correctness of your individual functions** and 2) according to the following criteria:

- 53 pts for correctly implementing the required functions
- 25 pts for correctly implementing the required classes
- 15 pts for correctly demonstrating your functions/classes in main
- 2 pts for using the correct file name and updating the starter code name/notes
- 5 pts for adherence to instructions

NOTE: you do not need to comment your code solutions.

Specifications

Write a program that completes the unimplemented sections of the deque class. Use the starter code as your starting place → make sure to create a new file following the required naming convention. **(DOWNLOAD the starter file FROM CANVAS – do not copy and paste)**

You have been provided a series of output statements in your main function in the starter code. Your main tasks after implementing the required functions and classes is to complete the main tasks using your derived class. Your output should match the provided example output.

(**15 pts**) Your main tasks are as follows (***see starter code main()***):

1. Declare instance of your Derived class with a size of 5 (this is 1 line of code)
2. Demonstrate class functions
    a. The first is the result of your abstract base class.
    b. The second demonstration is your overridden function.
3. Demonstrate deque functions
    a. Attempt to change deque size to 3
    b. Attempt to add 6 values to the deque using enqueue
        i. values are 2, 10, 20, 30, 40, 50 in that order
    c. Output number of items added successfully and display in forward order (front to back)
    d. Output value removed from FRONT, size of deque, and display in forward order (front to back)
    e. Output value removed from BACK, size of deque, and display in forward order (front to back)
    f. Display list in reverse order
    g. Output value added to FRONT, size of deque, and display in forward order (front to back)
    h. Output value added to FRONT, size of deque, and display in forward order (front to back)
    i. Display list in reverse order
    j. Display list in forward order
    k. Demonstrate printNthValue() function with positions 3, 2, and 20

Additional Requirements

- Use the given function prototypes and their formal parameters (*see starter code*) to implement the function definitions.
    - You are required to use implement the given functions and should not require additional functions to achieve the given tasks.

See the required functions/classes overview below and in the starter code.

Required Functions & Classes

You must implement the following 11 member functions **AND the destructor and constructor** as part of the `DynamicDeque` class. *Note that Your constructor, destructor, and isFull() have **stub functions** in the starter code already.*

- (**2 pts**) `DynamicDeque();`
    - `Initializes DynamicDeque variables`
- (**3 pts**) `~DynamicDeque ();`
    - `Removes all nodes and frees memory used in object instance`
- (**10 pts**) `void enqueueFront(int);`
    - `Adds an item to the front of the deque`
- (**10 pts**) `void dequeueBack(int&);`
    - `Removes an item from the back of the deque`

- (**2 pts**) `void displayForward(void) const;`
  - `Must be done using recursion`
  - `Displays all nodes in forward order (front to back)`
- (**3 pts**) `void displayForward(DequeNode*) const;`
  - `Overloaded helper function for displayForward`
- (**2 pts**) `void displayReverse(void) const;`
  - `Must be done using recursion`
  - `Displays all nodes in reverse order (back to front)`
- (**3 pts**) `void displayReverse(DequeNode*) const;`
  - `Overloaded helper function for displayReverse`
- (**2 pts**) `int getNumItems(void) const;`
  - `Returns the current size of the deque (aka number of nodes)`
- (**2 pts**) `bool isFull(void) const;`
  - `Returns true if the number of nodes is equal to the maximum size`
- (**4 pts**) `void resize(int);`
  - `Increases the maximum number of nodes (cannot decrease)`
  - `Outputs an error message if the new size is invalid`
- (**2 pts**) `void printNthValue(int) const;`
  - `Must be done using recursion`
  - `Position is counted from the front`
  - `Prints the value at the Nth node, e.g., N = 1 prints the 1st node value (so it is not a zero-based index but the position)`
- (**8 pts**) `void printNthValue(DequeNode*, int) const;`
  - `Overloaded helper function for printNthValue`

You are also required to implement 2 Classes as outlined below (and in the starter code)

- (**10 pts**) Class1
  - Create an abstract base class called `Generic` (you choose the method)
  - Create a public member function which cannot be overridden in a derived class
    - a. outputs "I cannot be overridden!"
  - Make your `DynamicDeque` class a public member (named deque) of the Generic class using composition/aggregation
- (**15 pts**) Class2
  - Create a publicly derived class named "`Derived`" from the abstract base class (in #1)
  - Create a default constructor that allows for setting the deque `maxSize` when constructing (Base class constructor sets to 0).
  - Declare an instance of your derived class variable in your main and specify a size of 5 using the constructor created in the previous step
  - Call the non-overridable public member function from main.
  - Complete the missing steps in main using your derived class object.

Expected Console Output

(note: the first line was my chosen way to demonstrate the abstract base class)

```
I was forced into existence by the abstract base class!
I cannot be overridden!

Error, deque can only be increased in size. Size unchanged. 3 <= 5.
The deque is full, 50 not added!
Added 5 items to the deque.
2 10 20 30 40

Removed a 2 from the front of the deque
The deque now contains 4 items.
10 20 30 40

Removed a 40 from the back of the deque
The deque now contains 3 items.
10 20 30

Now displaying the deque in reverse order (back to front)
30 20 10

Now adding a 42 to the front of the deque
The deque now contains 4 items.
42 10 20 30

Now adding a 9001 to the front of the deque
The deque now contains 5 items.
9001 42 10 20 30

Now displaying the deque in reverse order (back to front)
30 20 10 42 9001

The final deque now contains 5 items.
9001 42 10 20 30

Demonstrating printNthValue():
The value at position 3 is 10
The value at position 2 is 42
20 is an invalid position in this deque!
```