CS2308 - Foundations of Computer Science II
Program Assignment #5
Total Points: 100

Due 11:59 pm on April 30, 2020

This project contains two parts. In the first part, you will need to implement a **Dynamic Stack Template** class and a **Dynamic Queue Template** class. In the second part, you will solve the **File Reverser** problem and the **File Filter** problem by utilizing your Stack Template and Queue Template respectively.

Part 1:

Your Dynamic Stack Template class should have the following data and member functions.

```
template <class T>
class DynStack
{
        private:
                struct StackNode
                {
                        T value;
                        StackNode *next;
                };

                StackNode *top;

        public:
                DynStack();
                ~DynStack();
                void push(T);
                void pop(T &);
                bool isEmpty();
};
```

Your Dynamic Queue Template class should have the following data and member functions.

```
template <class T>
class Dynque
{
        private:
                struct QueueNode
                {
                        T value;
                        QueueNode *next;
                };

                QueueNode *front;
                QueueNode *rear;
                int numItems;
```

```
    public:
        Dynque();
        ~Dynque();
        void enqueue(T);
        void dequeue(T &);
        bool isEmpty();
};
```

Part 2:

File Reverser: Write a program (named as *file_reverser.cpp*) that opens the ***input.txt*** file and reads its contents into a stack of characters. The program should then pop the characters from the stack and save them in the ***output_reverse.txt*** file. The order of the characters saved in ***output_reverse.txt*** should be the reverse of their order in the ***input.txt*** file. **You must use the Dynamic Stack Template class you created in part 1 to solve the problem.**

File Filter: Write a program (named as *file_filter.cpp*) that opens the ***input.txt*** file and reads its contents into a queue of characters. The program should then dequeue each character, convert it to uppercase, and store it in the ***output_filter.txt*** file. The order of the characters saved in ***output_filter.txt*** should be the same as their order in the ***input.txt*** file, but all in uppercase. **You must use the Dynamic Queue Template class you created in part 1 to solve the problem.**

Sample Output:

input.txt: This is the original file.
output_reverse.txt: .elif lanigiro eht si sihT
output_filter.txt: THIS IS THE ORIGINAL FILE.

Suggested Timeline:

Apr. 21: Dynamic Stack Template class done.
Apr. 23: Dynamic Queue Template class done.
Apr. 27: File Reverser done.
Apr. 29: File Filter done.
Apr. 30: Due Date. Your project should be fully tested and commented.

Comments and Suggestions:

DO NOT DELAY. Start the project from Day 1. If you wait until the night before the due date, you will have a miserable night and it is less likely you could complete the project.

Do not try and write the entire program all at one time. Work on the program in small sections, as the timeline indicates.

Notes:

- Your program must correctly compile and be executable on the Linux server.
- Be sure your code files follow the format of the coding standard.

Program Submission

Please submit only your source file (firstname_lastname_prog5.zip) to TRACS. Your zip file should contain the following files:

- DynStack.h and DynStack.cpp (you can also implement them in one file).
- Dynqueue.h and Dynqueue.cpp (you can also implement them in one file).
- file_reverser.cpp
- file_filter.cpp

You will get zero if you fail to submit your project to TRACS before the deadline.

Your program will be graded as follows, please make sure to check each item before you submit.

CS2308 Program #5 Name:_____

Program and Run Time Output: _____ (90 Points)

_____ (20) Correctly implement the Dynamic Stack Template class

- _____ (2) Correctly implement the constructor function
- _____ (6) Correctly implement the push function
- _____ (6) Correctly implement the pop function
- _____ (3) Correctly implement the isEmpty function
- _____ (3) Correctly implement the destructor function

_____ (20) Correctly implement the Dynamic Queue Template class

- _____ (2) Correctly implement the constructor function
- _____ (6) Correctly implement the enqueue function
- _____ (6) Correctly implement the dequeue function
- _____ (3) Correctly implement the isEmpty function
- _____ (3) Correctly implement the destructor function

_____ (25) Correctly implement the File Reverser

- _____ (5) Correctly use the Dynamic Stack Template class to create a char stack
- _____ (10) Correctly use the char stack operations to read and store characters
- _____ (10) Correct file operations and single character get() and put() function

_____ (25) Correctly implement the File Filter

- _____ (5) Correctly use the Dynamic Queue Template class to create a char queue
- _____ (10) Correctly use the char queue operations to read and store characters
- _____ (10) Correct file operations, single character get() and put() and toupper() functions

Coding Standards: _____ (10 Points)

- _____ (3) Documentation (program and function headers)
- _____ (3) Comments
- _____ (2) Meaningful Variable Names
- _____ (2) Indentation Scheme / Use of { }

Total: _____ (100 Points)

Executable Version: _____ (%)

If your code cannot be compiled or executed, your final score will only be 50% of your accumulated score of each item above.

Final Score: _____ (100 Points)