CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 4, Sep 2018

Stacks and Queues

OBJECTIVES

1. Create, add to, delete from, and work with a stack implemented as a linked-list
2. Create, add to, delete from, and work with a stack implemented as an array
3. Create, add to, delete from, and work with a queue implemented as a linked-list
4. Create, add to, delete from, and work with a queue implemented as an array

Background

Stacks and Queues are both data structures that can be implemented using either an array or a linked list. To fully understand how they work in these different implementations, you will write both a Stack and a Queue using both an array and a linked-list.

Assignment

You will be building a todo list of items that will either go on to a stack or in a queue. A todo item for all implementations will be represented by the following struct:

```
struct TodoItem {
  string todo;
};
```

You will be implementing each of these data structures as its own class. The header files for all four implementations can be found on Moodle.

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 4, Sep 2018

Specifics

- Use the header files provided as definitions the classes for each data structure. Do not modify these files. Write one corresponding C++ file for each header file that implements all of the functions declared in that header file (a few of the getter functions are defined already, so you don't have to implement them). The names of these files are below:
    - *Provided Starter Code (do not submit):*
        - HW4-Todo-StackArray.hpp
        - HW4-Todo-StackLinkedList.hpp
        - HW4-Todo-QueueArray.hpp
        - HW4-Todo-QueueLinkedList.hpp
    - *Files you create (do submit):*
        - HW4-Todo-StackArray.cpp
        - HW4-Todo-StackLinkedList.cpp
        - HW4-Todo-QueueArray.cpp
        - HW4-Todo-QueueLinkedList.cpp

- **Do not** submit a main function as part of the implementations of each data structure. We are not looking for you to write stand-alone program as we did in the past - instead this assignment asks you to implement a library of code that can be imported wherever it is needed.

- To run and test your code, you should write a main function in a separate file. Use this main function to test your data structure. Because it's not part of the submission, we have no requirements for its behavior, but we suggest you exhaustively test all the functions you've written. To compile the file with your main function together with the file with your data structure functions, you should #include the corresponding header in both files, as well as compile both files together in g++ with a command similar to the one below (note the two .cpp files):

```
g++ --std=c++11 HW4-Todo-StackArray.cpp main.cpp -o hw4Test
```

- When working with array-based implementations, there is a max size available that is set to 5 for this assignment in the header files. You should print the following error when attempting to add to a full array:

```
"Stack full, cannot add new todo item."
```

or

```
"Queue full, cannot add new todo item."
```

Note - there is no maximum size for the linked-list implementations

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 4, Sep 2018

- If the stack or queue is empty when you try to pop or peek it, print the appropriate error message:

  ```
  "Stack empty, cannot pop an item."
  ```

  or

  ```
  "Stack empty, cannot peek."
  ```

  or

  ```
  "Queue empty, cannot dequeue an item."
  ```

  or

  ```
  "Queue empty, cannot peek."
  ```

  The return value when peeking from an empty list is undefined! Your code is allowed to return anything, and it will not be tested. This is the same behavior that the built-in standard library uses.

Submitting your code:

Submit your code on Moodle by following the directions on Assignment 4 Submit. You must submit an answer to be eligible for interview grading!