CSCI-2270
Assignment 4
Instructor: Boese

Stacks and Queues

Objectives

- Create, add and delete, and work with a stack using a linked-list
- Create, add and delete, and work with a stack using an array
- Create, add and delete, and work with a queue using a linked-list
- Create, add and delete, and work with a queue using an array

---

Background

Stacks and Queues are both data structures that can be implemented using either an array or a linked list. Therefore, to fully understand how they work you will complete the implementation for both a Stack and a Queue using both an array and a linked-list.

Assignment

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

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

You will also be working with classes in this assignment. The code for header files for all four implementations are provided in this write-up.

Program Specifications

- Use the starter code in this write-up to create the following header files. Do not modify what is provided. You will write the implementation files and submit those only.

  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 add a main method to any of your submitted files.
- DO write your own test drivers to test your code, but do not submit them.
- Your code needs to be readable, efficient, and accomplish the task provided.
- Make sure you delete your dynamically allocated memory in the appropriate methods!
- When working with array-based implementations, there is a max size available (set to 5 for this assignment in the header files). Display an error message if attempting to add to a full array:
  "Stack full, cannot add new todo item."
  Or "Queue full, cannot add new todo item."
  Note – this does not apply to linked-list implementations.
- If the stack or queue is empty when you try to pop or peek it, display an error message:
  "Stack empty, cannot pop an item."
  "Stack empty, cannot peek."
  "Queue empty, cannot dequeue an item."
  "Queue empty, cannot peek."
- Make sure your code is commented enough to describe what it is doing. Include a comment block at the top of all .cpp files with your name, assignment number, and course instructor, and anyone you worked with.
- You must implement the functions as specified. You do not need any additional functions. Each .hpp file has one or more getter methods that are defined in the header. You do not need to implement these.
- Use the following code for your header files, and name them as indicated. DO NOT MODIFY.

HW4-Todo-QueueArray.hpp

```
#ifndef HW4_TODO_QUEUEARRAY
#define HW4_TODO_QUEUEARRAY

#include <string>

struct TodoItem
{
        std::string todo;
};

const int MAX_QUEUE_SIZE = 5;

class TodoQueueArray
{
  public:
        TodoQueueArray();
        bool isEmpty();
        bool isFull();
```

```
        void enqueue(std::string todoItem);
        void dequeue();
        TodoItem* peek();
        int getQueueFront() { return queueFront; }
        int getQueueEnd()   { return queueEnd; }
        TodoItem** getQueue() { return queue; }

  private:
        int queueFront; //the index in queue[] that will be dequeued next
        int queueEnd;   //the index in queue[] that was most recently enqueued
        TodoItem* queue[MAX_QUEUE_SIZE];
};

#endif
```

HW4-Todo-QueueLinkedList.hpp

```
#ifndef HW4_TODO_QUEUELINKEDLIST
#define HW4_TODO_QUEUELINKEDLIST

#include <string>

struct TodoItem
{
        std::string todo;
        TodoItem *next;
};

class TodoQueueLinkedList
{
  public:
        TodoQueueLinkedList();
        ~TodoQueueLinkedList();
        bool isEmpty();
        void enqueue(std::string todoItem);
        void dequeue();
        TodoItem* peek();
        TodoItem* getQueueFront() { return queueFront; }
        TodoItem* getQueueEnd()       { return queueEnd; }

  private:
        TodoItem* queueFront; // the item in the list that will be dequeued next
        TodoItem* queueEnd;   // the item in the list that was most recently enqueued
};

#endif
```

HW4-Todo-StackArray.hpp

```
#ifndef HW4_TODO_STACKARRAY
#define HW4_TODO_STACKARRAY

#include <string>

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

```
const int MAX_STACK_SIZE = 5;

class TodoStackArray
{
  public:
          TodoStackArray();
          bool isEmpty();
          bool isFull();
          void push(std::string todoItem);
          void pop();
          TodoItem* peek();
          int getStackTop()    { return stackTop; }
          TodoItem** getStack() { return stack; }

  private:
          int stackTop;       //the index in stack[] that will be popped next
          TodoItem* stack[MAX_STACK_SIZE];
};

#endif
```

HW4-Todo-StackLinkedList.hpp

```
#ifndef HW4_TODO_STACKLINKEDLIST
#define HW4_TODO_STACKLINKEDLIST

#include <string>

struct TodoItem
{
          std::string todo;
          TodoItem *next;
};

class TodoStackLinkedList
{
  public:
          TodoStackLinkedList();
          ~TodoStackLinkedList();
          bool isEmpty();
          void push(std::string todoItem);
          void pop();
          TodoItem* peek();
          TodoItem* getStackHead()      { return stackHead; }

  private:
          TodoItem* stackHead; // pointer to the TodoItem that will be popped next
};

#endif
```

<u>Example to-do items could include:</u>
Take out the garbage

Clean the dishes
Make bed
Do laundry
Buy detergent

Submitting Your Code:

Log into Moodle and go to the Homework 4 link. It is set up in quiz format. Follow the instructions on each question to submit all or parts of your code. You can check your solution to each question by clicking on the "Check" button multiple times without penalty. Note that you can only **submit** your homework once.

Note: there is no late period on assignments! If you miss the deadline or do not do well, you can sign up for an optional grading interview to get up to half the points missed back. There is also an optional extra credit assignment at the end of the semester you can use to replace one of your homework scores.