ECGR 3180 Fall 2020
Dr. Mihail Cutitaru

Assignment 3
Assigned: October 15, 2020
Due: October 22, 2020 by 11:59PM in Canvas

1. **(40 pts)** Fill in the functions for the code snippet below. The functions will implement an integer stack using deques only. Try to implement it using only one deque, but a solution with two deques will also be accepted. You may add additional public helper functions or private members/functions.
   Do not submit your test code (**`main()`** function), but your functions need to account for test cases (e.g. pop when the stack is empty, etc).
   What is the Big-Oh ***O()*** time for your implementation of the functions **`push()`** and **`pop()`** below? Assume there are *n* elements in stack.

   The list of available options for a deque is available at: http://cplusplus.com/reference/deque/deque/
   Your MyStack class will be instantiated via a pointer and called as shown below:

```
MyStack *ptr = new MyStack();
ptr->push(value);
int pop1 = ptr->pop();
int pop2 = ptr->pop();
bool isEmpty = ptr->empty();
```

   Name the file you submit **`MyStackDeque.cpp`**

```
class MyStack {
public:
    // Default Constructor
    MyStack() {// ... }

    // Push integer n onto top of stack
    void push(int n) {// ... }

    // Pop element on top of stack
    int pop() {// ... }

    // Get the top element but do not pop it (peek at top of stack)
    int top() {// ... }

    // Return whether the stack is empty or not
    bool empty() {// ... }
};
```

2. **(10 pts)** Implement the **`MyStack`** class in Problem 1 using vectors. Name the file you submit **`MyStackVector.cpp`**

3. **(50 pts)** Fill in the functions for the code snippet below. The functions will implement an integer list using a dynamic array only (an array that can grow and shrink as needed, uses a pointer and size of array.) For more information: https://www.geeksforgeeks.org/how-do-dynamic-arrays-work/
   You may add additional public helper functions or private members/functions.
   Do not submit your test code (**`main()`** function), but your functions need to account for test cases (e.g. pop when the list is empty, etc).
   Your **`MyList`** class will be instantiated via a pointer and called similar to the code in Problem 1.
   Name the file you submit **`MyListDyn.cpp`**

```
class MyList {
public:
    // Default Constructor
    MyList() {// ... }

    // Push integer n onto the end of the list
    void push_back(int n) {// ... }

    // Push integer n onto the beginning of the list
    void push_front(int n) {// ... }

    // Pop element at the end of list
    int pop_back() {// ... }

    // Pop element at the beginning of list
    int pop_front() {// ... }

    // Adds value at index pos. Indices start at 0
    void emplace(int pos, int value) {// ... }

    // Return whether the list is empty or not
    bool empty() {// ... }
};
```