Problem Set 2: Lists, Stacks, and Queues

CS3330 Data Structures and Algorithms
Term 5 2016: May 23 – July 24
Dr. Jack Davault

Overview: For problems 1(a) and 2(a) of this assignment, you will need a C++ compiler. In order to receive credit, your programs must compile and run; and you must provide the actual source code file so that I can compile and run your programs (e.g. the files ending in **.cpp**). The remaining problems for the assignment must be written up within a single Microsoft Word document. You must include your name and course number within all files that you submit, including source code files as a comment at the top of each file that you modify.

Problem 1. [5 points] Arrays and Linked Lists: Read the assigned chapters and notes for Module 3 located in the Learning Activities area. Then provide solutions to the following:

(a) [3 points] Download the **List.zip** file and then modify **List.cpp** to implement details for the predicate function called `isInList()`. The prototype for this function is as follows:

```
template <class Object>
bool isInList(Object element, list<Object> myList);
```

The implementation will use the C++ Standard Template Library **list** implementation. The function will determine if a given item (`element`) is in a given list (`myList`) and will return `true` (or 1) if it is and `false` (or 0) if it is not.

Hint: You only need to modify the areas of the code in the **List.cpp** file that are marked with the TODO comments. Everything else will remain that same. You can implement this function anyway you like, but you must use the provided list and function prototype.

Output: Once the function is implemented, the output for the program will appear as follows:

```
myList:
 282 471 786 277 646 832 571 792 567 348

172 is in list (0=false, 1=true): 0
178 is in list (0=false, 1=true): 0
282 is in list (0=false, 1=true): 1
471 is in list (0=false, 1=true): 1
571 is in list (0=false, 1=true): 1
775 is in list (0=false, 1=true): 0

** Press any key to continue **
```

(b) [2 points] Write a brief summary explaining the differences between an array-based list and a linked list data structure. Describe the advantages and disadvantages of both types of lists in comparison to each other.

Problem 2. [5 points] Stacks and Queues: Read the assigned chapters and notes for Module 4 located in the Learning Activities area, then provide solutions to the following:

(a) [3 points] Download the **Stack.cpp** file. Modify the to implement details for the helper function called `reverseStack()`. The prototype for this function is as follows:

```
template <class Object>
void reverseStack(stack<Object> myStack);
```

Again, this implementation will use the C++ Standard Template Library. Specifically the stack API. The function take a stack called `myStack` as input, and will reverse the items and store the results back into original stack variable.

Hint: Again, you only need to modify the areas of the code in the **Stack.cpp** file that are marked with TODO comments. Everything else in the program should remain that same. Use the stack API's `top()`, `pop()`, and `push()` functions. Consider using a temporary stack to temporarily hold the results. For example, declare a temporary stack first:

```
stack<Object> tmpStack;
```

The while-loop can walk through the main stack and call the stack `top()` and `pop()` operations on the main stack list the `printStack()` function. Add each item to the temporary stack by calling the `push()` operation on the main stack. For example:

```
tmpStack.push(myStack.top());
```

After the while-loop set `myStack` equal to the temporary stack.

Output: The output for the program after the function is implemented should appear as follows:

```
The contents of myStack:
 582
 *
 346
 +
 259
 -
 457
 *
 462
 -
 423

The reverseStack of myStack:
 423
 -
 462
 *
 457
 -
 259
 +
 346
 *
 582

** Press any key to continue **
```

(b) [2 points] When considering the `push()` and `pop()` operations for a stack, briefly describe how a compiler program could use a stack to implement delimiter matching. For example, matching delimiter strings could be: “{”, “}”, “(”, “)”, and “/*”, and “*/”.

Other Notes: Submit your solutions as a single Zip file using the Problem Set 2 link provided in the Assignments area. If you are using the Visual C++ or Dev-C++ compiler, only submit the source code files for your program (the files ending in **.cpp**). For space reasons, please do not submit the entire Visual C++ or Dev-C++ project folders. Let me know if you have any questions or need clarification on what to do for this assignment.