Programming Assignment 1

Data Science

United International College

Rubrics

- Refer to the Rubrics for programming on iSpace
- You will get full mark for Function test if
  - Your code produces correct output for all our test inputs (hidden).
    - The test inputs are not provided to you.
    - Try your code against all possible inputs (that you can think of) to test correctness
  - No memory leak is found in any case
- Program Structure refers to
  - Reasonable file structure in the project
  - Reasonable placement of function declarations and implementations
- Code style includes
  - Reasonable naming of identifiers
  - Reasonable indentation
  - Code neatness

Task1: List Methods

- Given the Linked list ADT introduced in Lecture 3, implement three more methods:
  - `DeleteNodes`
  - `RemoveDuplicates`
  - `ReverseList`
- Submit the complete code set including
  - Struct definition
  - Declaration and implementation for the existing and the new methods
    - You may use the sample solution provided on iSpace or your own implementation of the existing methods (e.g., *InsertNode*).
  - A main function which runs your own test cases.

DeleteNodes

- int DeleteNodes(Node** phead, double x);
  - Removes all the elements from a linked list that have value *x*.
  - Returns the number of occurrences of *x.*
- Sample Input and output:

Input	List Update	Returned Value
2 -> 6 -> 5 -> 6 -> NULL, x = 6	2 -> 5-> NULL	2
NULL, x = 6	NULL	0
6 -> 6 -> 6 -> NULL, x = 6	NULL	3

RemoveDuplicates

- void RemoveDuplicates(Node** phead);
  - Remove duplicates from sorted list (you may assume that the node values in the list are in non-decremental order)
  - Deletes all nodes that have duplicate values, leaving only nodes with distinct values
  - Your implementation should finish the removing with SINGLE traversal of the whole list, which means you cannot simply invoke the DeleteNodes method for removing the duplicates.
- Sample Input and output:

Input	**List Update**
1 -> 2 -> 2 -> 4 -> 6 -> 6 -> NULL	1 -> 4-> NULL
NULL	NULL
6 -> 6 -> 6 -> 7 -> 7 -> NULL	NULL

ReverseList

- void ReverseList(Node** phead);
  - Reverse the linked list WITHOUT using extra space.
  - Hint: Reverse the linked list can be performed by modifying the “next” pointer of current node from the original next node to its previous node.
- Sample Input and output:

Input	**List Update**
1 -> 2 -> 3 -> 4 -> 5 ->NULL	5->4->3->2->1->NULL
NULL	NULL
1->NULL	1->NULL

Task 2: ValidBrackets

- Complete function: bool ValidBrackets(char* str)
  - str is a string containing only ‘(’, ‘)’, ‘{’, ‘}’, ‘[’ and ‘]’.
  - Returns True if the input string is valid and False otherwise
  - In a valid string,
    - The brackets must match
    - The brackets must close in the correct order
- Sample Input and output

Input	Output
"{()}"	True
"([)]"	False
"{()}["	False
""	True
NULL	False

Task 2: ValidBrackets

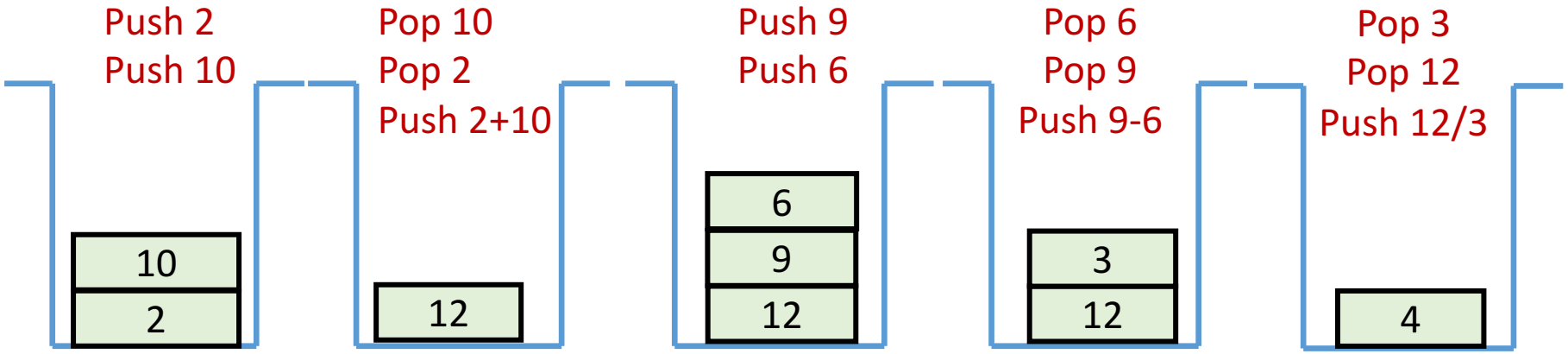
- Hint
  - You can make use of the Stack ADT, assuming the length of str will not exceed 50.
  - Consider what action you will take when you process the following characters in the string
    - ‘{‘, ‘[‘, ‘(‘: opening brackets
    - ‘}’, ’]’, ‘)’: closing brackets
- Submit the complete code set including
  - Struct definition
  - Declaration and implementation for every necessary method
  - A main function which runs your own test cases

Task 3: Postfix Expression Evaluation

- Consider the postfix expression evaluation algorithm introduced in Lec3:
  - If the element is an operand, push it to stack
  - If the element is an operator O, pop twice and get A and B respectively. Calculate BOA and push it back to stack
  - When the expression is ended, the value in the stack is the final answer.
- Complete function: int postfixEval(char* postfix)
  - postfix is a string representing a valid postfix expression which contains only digits and '+', '-', '*', '/' operators, separated by space.
    - Hint: Use strtok function for splitting a string by some delimiter
  - You can assume the length of postfix will not exceed 50.
  - Returns the evaluation result.

Task 3: Postfix Expression Evaluation

- Submit the complete code set including
  - Struct definition
  - Declaration and implementation for every necessary method
  - A main function which runs your own test cases
- Sample input and output:
  - Postfix expression: 9 1 3 * / , result is 3.
  - Postfix expression: 2 10 + 9 6 - /, result is 4 (as illustrated in the Figure).



Submission

1. Put the complete set of source files for each problem into a folder named with the problem ID.
   - For example, the code set (.h and .cpp files) for Problem 1 should be in folder 1.
2. Compress all the folders into a zip with name: PA1_<your student id>.zip
3. Submit the .zip file to iSpace.

Plagiarism Policy

- You are encouraged to collaborate in study groups.
  - But, you cannot copy or slightly change other students' solutions or codes.
- We will check between everyone's submission.
- We will check with online solutions.
- If copies are found, everyone involved gets ZERO mark.