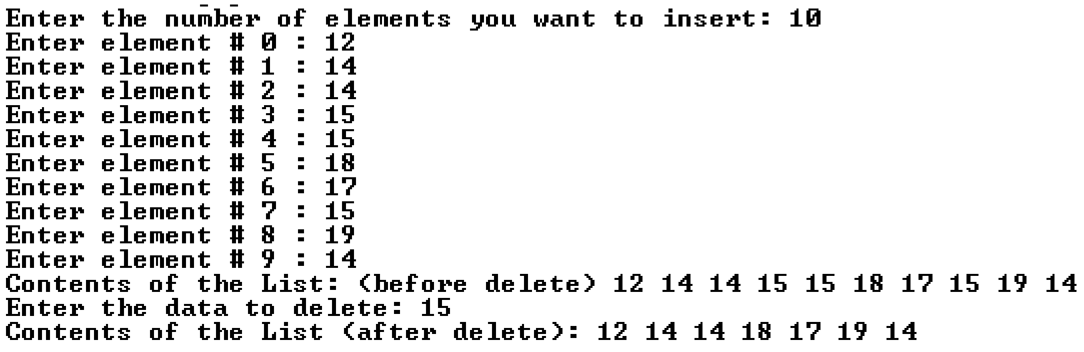
CSC 228 Data Structures and Algorithms, Fall 2017
Instructor: Dr. Natarajan Meghanathan

Exam 1 (Take Home) **Due: Oct. 2, 2017: 1 PM (submission through Canvas)**

Q1 - 25 pts) The deleteElement(int deleteData) member function in the code for the Singly Linked List-based implementation of a List ADT deletes the first occurrence of deleteData in the List. Modify this member function in such a way that all occurrences of deleteData in the List are deleted with a single call to the deleteElement function from main. After you modify the *deleteElement* function, run the main function (as given in the startup code for this question) by creating a List of at least 10 elements with certain elements repeating. Now ask for a value (deleteData) to delete from the user and call the deleteElement(deleteData) function to delete all occurrences of deleteData in the List. Capture the output of your program displaying the contents of the List before and after the call to the deleteElement function.

A sample of the expected screenshot of the program is shown below. As noticed in the screenshot, all occurrences of deleteData '15' in the List are removed after a call to the deleteElement function.


```
Enter the number of elements you want to insert: 10
Enter element # 0 : 12
Enter element # 1 : 14
Enter element # 2 : 14
Enter element # 3 : 15
Enter element # 4 : 15
Enter element # 5 : 18
Enter element # 6 : 17
Enter element # 7 : 15
Enter element # 8 : 19
Enter element # 9 : 14
Contents of the List: (before delete) 12 14 14 15 15 18 17 15 19 14
Enter the data to delete: 15
Contents of the List (after delete): 12 14 14 18 17 19 14
```

Q2 - 25 pts) The Fibonacci sequence is generated using the following recursion: F(n) = F(n-2) + F(n-1), for n > 1. F(0) = 0 and F(1) = 1.
In this question, you will generate the Fibonacci sequence as a "Singly Linked List" of integers 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ..., N where N is the largest integer in the sequence that is less than the integer J, comprising the last three digits of your J#.
For example, if your J# is J00543244, then J is 244. In that case, the Fibonacci sequence that is generated should be 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233.
If the last three digits of your J# form an integer that is less than 100, add 100 to the last three digits of your J# and use the resulting value as J. For example, if your J# is J00543034, then J is 34 + 100 = 134 and the Fibonacci sequence generated would be: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.

You are given the code for the Singly Linked List-based implementation of the List ADT. Add a member function constructSequence(int) to the List class that takes an integer argument (*upperBound*: J for the Fibonacci sequence)

The main function given to you already inserts the first two nodes (with data 0 and 1 respectively) to a List called FibonacciList. Your task will be to implement the *constructSequence* function that will insert a sequence of nodes (one node per iteration) to the FibonacciList whose last element will be the largest element in the sequence that is less than the *upperBound.*

Q3 - 20 pts) You are given the code (including the main function) for evaluating an expression in post-fix format using Stack. Your task is to modify the code in the main function to input an expression in pre-fix format, reverse it (as discussed in class) and then evaluate the value of the reversed expression (scanned from left to right) using Stack.

You test your code with an expression string in pre-fix format that comprises of all the four operators (at least once) in a randomly chosen order with randomly chosen values in the range 1 to 9. For example, a sample expression string (pre-fix notation) could be input as **-, +, *, /, 5, 2, 3, 4, 8** for the expression (infix-notation) 5 / 2 * 3 + 4 - 8.

Q4 - 30 pts) Implement the **Stack ADT as a Singly Linked List** of integers. You will implement the push(int), pop( ), peek( ) and isEmpty( ) functions of the Stack. *All the functions operate with the node at the beginning of the List.* So, an element pushed to the Stack will be inserted as the first element in the List (i.e., as the next node of the head node). The peek operation will read the value of the first element in the List, whereas the pop operation will delete the first element from the List. The isEmpty function will check whether the next node of the head node is NULL or not. You will test your code with a main function that inputs 5 integers from the user (or generated randomly), pushes each of them to stack, reads/prints the top value of the Stack after pushing the 5 integers and finally pops all the elements of the Stack as well as prints the value of the integers popped.

For reference, you are given the code for the implementation of Stack as a Doubly Linked List.

Submission (through Canvas):
Submit a single word document that contains the complete C++/Java code (including the main function for each question). You should also include the screenshots as required for the output of each question. Clearly label the question numbers for each code.