Sample Past Test 2 Questions

Name: **CS 3358** **Grade:**

Use the following `Node` ***definition for all linked list related problems:***

```
struct Node
{
   typedef int Item;
   Item data;
   Node *link;
};
```

---

1. Suppose you are an instructor for a class and want to keep track of a list of test scores in order. You are thinking about using the **SortedList** ADT that you implemented in Homework 2 (using *array*) and Homework 3 (using *linked list*) in a program.

   (a) As you grade each test, you insert the score into the list. In other words, you build the list as you grade the tests. Which of the two **SortedList** implementations mentioned above would be more efficient for building the list? Why? (Be brief and concise but clear.)

   (b) Now suppose that you already have such a sorted list of test scores. Students keep asking questions of the form "What is the 5th highest score in the class?", or, in general, "What is the *i*th highest score in the class?". Which of the two **SortedList** implementations mentioned above would be more efficient for answering such questions? Why? (Be brief and concise but clear.)

2. Write a C++ function called `IsEqual` that can be used to determine if two (singly) linked lists are equal to each other. Two lists are equal if they contain the ***same number of nodes*** and their nodes contain the ***same values in the same order***. The function receives the head pointers of the two lists and returns true if the lists are equal and false otherwise. Do ***not*** use functions defined in the linked list toolkit.

3. Given a pointer to a node (**`givenNodePtr`**) in a singly linked list that is not the head node or the tail node, write pseudocode for an algorithm (or write a C++ function) that can be used to swap the node with its immediate successor node. You can assume that the linked list has at least three nodes. You are to do this ***only by adjusting the pointers*** (*i.e.*, do not adjust the data, create any new node or use any functions in the linked list toolkit).

4. Use a stack to convert the following expression from infix to postfix. Then use another stack to evaluate the expression. In each process, indicate the content of the stack after each step.

   `INFIX expression: (2 + 3 * 4) * (5 * (7 - 6) + 9) - 8`

   `INFIX to POSTFIX:`
   `(Note: show contents of stack so that` ***`top is to the right`***`)`

```
Symbol  Stack                    Output
------  -----                    ------
```

POSTFIX expression:

EVALUATE POSTFIX:
(Note: show contents of stack so that ***top is to the right***)

```
Symbol  Stack                            Remarks
------  -----                            --------
```

5. Write pseudocode for an algorithm that uses ***one or more stacks*** (***no*** queues) to determine whether an input of a string of characters forms a *palindrome* (*i.e.*, whether the sequence of characters is spelled the same forward and backward). *Non-alphabet* characters and *case* of characters are to be ignored in the palindrome-recognizing process.

6. What will be printed by the following program?

```
#include <iostream.h>
#include <stdlib.h>

void Mystery(int num);

int main()
{
   int num = 1234;

   Mystery(num);

   return EXIT_SUCCESS;
}

void Mystery(int num)
{
  int x, y;
  if (num > 0)
  {
    x = num % 10;
    cout << x;
    Mystery(num / 10);
  }
  y = num % 10;
  cout << y;
}
```