CSC 220 Midterm Sample Solutions

Sample Questions

1. What does the following code **print to the screen**? You may assume **Stack** is an existing implementation of a Stack.

```
public static void main() {
      Stack numbers = new Stack(5);

      numbers.add(10);
      numbers.add(20);
      numbers.add(30);
      numbers.add(40);

      System.out.println(numbers.remove());

      numbers.add(50);

      while (numbers.empty() != true) {
            System.out.println(numbers.remove());
      }
}
```

Solution:
40
50
30
20
10

2. Write a class which stores a private array of 10 random integers. Write a function in this class which returns the average of the values in the array. Use the function signature **int average();**

Solution:
I am using the Random class in this solution to generate random numbers. On the exam, any method for loading the array with values will be acceptable.

```java
import java.util.Random;

public class Q2 {
  private int[] numbers = new int[10];

  public Q2() {
     Random rand = new Random();
     for (int i = 0; i < 10; i++) {
        numbers[i] = rand.nextInt(100);
     }
  }

  public int average() {
     int total = 0;
     for (int i = 0; i < numbers.length; i++) {
        total = total + numbers[i];
     }

     return total / numbers.length;
  }
}
```

3. Implement the **remove** (also called "**dequeue**") function of the Queue data structure. Remember that the Queue is a data structure which only allows data to be inserted at the beginning of the list (i.e. index 0 of an array) and removed from the end of the list (i.e. the last array index). Use the following class definition as a template:

```
class Queue {
     private int[] data;
     private int indexOfLastValueStored;

     public Queue() {
          data = new int[25];
          indexOfLastValueStored = 0;
     }
```

```
        int remove() {
            lastIndex--;
            int returnValue = data[lastIndex];

            return returnValue;
        }
}
```

4. Implement the **add** function of the Queue data structure. Remember that the Queue is a data structure which only allows data to be inserted at the beginning of the list (i.e. index 0 of an array) and removed from the end of the list (i.e. the last array index). The mnemonic often used to remember this is “first in, first out”. Use the following class definition as a template:

```
class Queue {
      private int[] data;
      private int indexOfLastValueStored;

      public Queue() {
            data = new int[25];
            indexOfLastValueStored = 0;
      }

      public void add(int newData) {
            // Shift data by 1 index to free up index 0
             for (int i = lastIndex; i >= 0; i--) {
                 data[i+1] = data[i];
             }

             // Insert data at index 0 (“beginning” of
list)
             data[0] = newNumber;

             lastIndex++;
      }
}
```

5. Write a program which prints a 2D array **numbers** in reverse order, from the bottom-right to the top-left. You may assume the array exists and is full of numbers already.

Solution:

```
for (int i = numbers.length; i > 0; i--) {
      for (int j = numbers[i].length; j > 0; j--) {
            System.out.print(numbers[i][j]);
      }

      // Add a new line to start the next row
      System.out.println();
}
```

6. Write a function which prints all the items stored in a Stack **in the order the items were inserted**. You may use **only** the **add** and **remove** functions which the Stack provides. Remember that the Stack only allows data to be added or removed from the end of the list. The last data point added to the Stack is the first data point to be removed. The mnemonic for this is concept is “last in, first out”.

You may assume the variable **stack** exists, it is not empty, and a function **size()** is defined, which returns the current size of the stack.

Solution:
I included more code than required here so the solution will be easier to test for yourselves using one of the Stack implementations we built in class.

```
public class Main {
  public static void main(String[] args) {
    // Using the most recent implementation of the Stack,
    // which uses an array to store its data.
    Stack stack = new Stack(10);
    stack.add(5);
    stack.add(6);
    stack.add(7);
```

```
        stack.add(8);
        stack.add(9);
        stack.add(10);

        Stack temporaryStack = new Stack(stack.size());

        // Add numbers from the original "stack"
        // to the temporaryStack in reverse order
        while (stack.size() > 0) {
            int number = stack.remove();
            temporaryStack.add(number);
        }


        // Removing the numbers from the temporaryStack
        // reverses the order of the numbers again,
        // printing the numbers in the order they were added originally.
        while (temporaryStack.size() > 0) {
            int number = temporaryStack.remove();
            System.out.println(number);
        }
    }
}
```

7. Complete the function below which adds a new node to the end of a linked list. Use the Node object defined below to store data. For the purposes of this exercise, only the default constructor is defined and the member variables are public.

```
// Definition of the Node class
class Node {
      public int data;
      public Node next;

      public Node() {
            data = 0;
```

```
            next = null;
      }
}



// Practice Linked List
class LinkedList {
      public Node head;

      public LinkedList() {
            head = null;
      }

      public void insertNewNode() {
            if (empty()) {
                  head = node;
            } else {
                  Node currentNode = head;

                  while (currentNode.next != null) {
                        currentNode = currentNode.next;
                  }

                  currentNode.next = node;
             }
      }

      // Remainder of class omitted.

}
```