COE 212 – Engineering Programming

Welcome to the Final Exam
Thursday December 15, 2016

Instructors: Dr. Salim Haddad
Dr. Bachir Habib
Dr. Joe Tekli
Dr. Wissam F. Fawaz

Name: _____

Student ID: _____

Instructions:

1. This exam is **Closed Book**. Please do not forget to write your name and ID on the first page.
2. You have exactly **125 minutes** to complete the **6** required problems.
3. Read each problem carefully. If something appears ambiguous, please write your assumptions.
4. Do not get bogged-down on any one problem, you will have to work fast to complete this exam.
5. Put your answers in the space provided only. No other spaces will be graded or even looked at.

Good Luck!!

Problem I: Arrays (**20 minutes)** [20 points]

For the questions given below, consider the following Java program. Note that numbers are placed at the beginning of each statement to simplify the process of referencing these statements in the subsequent questions. So, the considered program compiles correctly and hence is syntactically valid.

```
1. import java.util.ArrayList;
2. public class ProblemI {
3.    public static void main(String[] args) {
4.          int[][] mat = {{1, 2, 3}, {4, 5, 6}};
5.          ArrayList<Integer> list = new ArrayList<>();
6.          int[] arr1 = new int[mat[0].length];
7.          int[] arr2 = new int[mat[1].length];
8.          for(int j=0; j<mat[0].length; j++)
9.                list.add(mat[0][j]);
10.         int index = 0;
11.         while(!list.isEmpty()) {
12.               arr1[index] = list.get(0);
13.               list.remove(0);
14.               index++;}
15.         for(int j=mat[1].length-1; j>=0; j--)
16.               list.add(mat[1][j]);
17.         int size = list.size();
18.         index = 0;
19.         for(int i=1; i<=size;i++) {
20.               arr2[index] = list.get(0);
21.               list.remove(0);
22.               index++;}
23.         for(int i=0; i<arr1.length; i++)
24.               System.out.print(arr1[i] + " ");
25.         System.out.println();
26.         for(int i=0; i<arr2.length; i++)
27.               System.out.print(arr2[i] + " ");}}
```

1) What is the value of the variable `size` created at line **number 17**?
   a. 2
   b. 6
   c. **3**
   d. None of the above
2) What output does the full execution of the `for` repetition statement at line **number 23** produce?
   a. `1 4`
   b. **`1 2 3`**
   c. `2 5`
   d. None of the above
3) What output does the full execution of the `for` repetition statement at line **number 26** produce?
   a. `3 6`
   b. `4 5 6`
   c. `2 5`
   d. **None of the above**
4) Which of the following can be used to compute the product of the elements of `arr1`?
   a. `int prod = arr1[1]*arr1[2]*arr1[3];`
   b. `int prod=1; for(int i=0;i<arr1.length();i++) prod*=arr1[i];`
   c. Both of the above
   d. **None of the above**

5) Which of the following can be used to find the minimum value in `arr2`?
   a. `int m=arr2[2]; for(int i=0;i<2;i++) if(m<arr2[i]) m=arr2[i];`
   b. `int m=arr2[0]; for(int i=0;i<3;i++) if(m<arr2[i]) m=arr2[i];`
   c. **Both of the above**
   d. None of the above
6) Which of the following can be used to swap the values of the first and second elements of the first row in `mat`?
   a. `mat[0][0] = mat[0][1];`
   b. `mat[1][1] = mat[1][2];`
   c. `mat[0][1] = mat[0][2];`
   d. **None of the above**
7) Which of the following correctly finds the maximum element in `mat`?
   a.
```
int m = mat[0][0];
for(int i=0;i<mat.length;i++)
for(int j=0; j<mat[i].length; j++)
      if(m<=mat[i][j]) m = mat[i][j];
```
   b.
```
int m = mat[mat.length-1][mat[0].length-1];
for(int i=0;i<mat.length;i++)
for(int j=0; j<mat.length; j++)
      if(m>mat[i][j]) m = mat[i][j];
```
   c. Both of the above
   d. None of the above
8) Which of the following correctly computes the sum of the elements of `mat`?
   a.
```
int sum = mat[0][0];
for(int i=1;i<mat.length;i++)
for(int j=1; j<mat[i].length; j++) sum+=mat[i][j];
```
   b.
```
int sum = 0;
for(int i=0;i<mat.length;i++)
for(int j=0; j<mat[i].length; j++) sum+=mat[i][j];
```
   c. Both of the above
   d. None of the above
9) Which of the following correctly checks whether the size of `list` is between `1`(inclusive) and `3` (exclusive)?
   a. `if(list.size >= 1 || list.size <= 3)`
   b. `if(list.size() >= 1 && list.size() <= 3)`
   c. `if(list.size() >= 1 && list.size() > 3)`
   d. **None of the above**
10) Which of the following can be used to empty `list`?
   a. `while(list.isEmpty()) list.get(0);`
   b. `while(!list.isEmpty()) list.get(0);`
   c. Both of the above
   d. **None of the above**

Problem II: True or false questions (**15 minutes)** [10 points]

1. Declaring a variable with `protected` visibility gives the variable package access, that is, only child classes can reference the variable by name.

Answer: True **False**

2. Overloading a method differs from overriding because overloaded methods have the same signature.

Answer: True **False**

3. A class wishing to override the implementation of the `toString` method must use the following header for the overridden method:
   `public String toString()`
   Answer: **True** False

4. Consider the following method definition.

```
public void method1(int[] arr) {
      int temp;
      temp = arr[0]; arr[0] = arr[1];
      arr[1] = temp;
}
```

   The following code fragment prints: `1 2 3 4 5`

```
int[] arr = {2, 1, 3, 4, 5};
method1(arr);
System.out.print(arr);
```

Answer: True **False**

5. Consider the following method definition. Calling `method2` with a parameter of `Z` (**capital Z**) returns a value `1`

```
public int method2(char val) {return 'a' - val;}
```

Answer: True **False**

6. Consider the following method definition. `method3` correctly checks whether or not the input character parameter `val` is a vowel

```
public void method3(String val) {
      String[] arr = {"a", "o", "e", "u", "i"};
      val = val.toLowerCase();
      for(int i=0; i<arr.length; i++)
            if(arr[i].equals(val))
                  return true;
      return false;}
```

Answer: **True** False

7. A subclass can call the constructor of its super class using the `super` reserved word

Answer: **True** False

8. The default version of `equals` determines whether two objects are aliases of each other

Answer: **True** False

9. A child can access all the variables declared in the parent class but the parent class cannot access the variables declared in the child class.

Answer: True **False**

10. A child class inherits all the methods of its parent class.

Answer: True **False**

Problem III: Code Analysis (**15 minutes**) [10 points]

1) What is the output of the following Java program?

```
public class Problem3a {
      public static void main(String[] args) {
            int count = 1;
            int[][] mat = new int[3][2];
            for(int i=0; i<mat.length; i++) {
                  for(int j=0; j<mat[i].length; j++) {
                        mat[i][j] = count;
                        count++;
                  }
            }
            int output = method1(mat);
            System.out.println("Output: " + output);}
      private static int method1(int[][] mat) {
            int prod, output = 0;
            for(int i=0; i<mat.length; i++) {
                  prod = 1;
                  for(int j=0; j<mat[i].length; j++)
                        prod *=mat[i][j];
                  output += prod;}
            return output;}
}
```

a. `Output: 44`
b. `Output: 52`
c. `Output: 35`
d. It does not compile correctly
e. None of the above

2) You are given below an interface, a class implementing the interface, and a driver class for the latter. What output is produced when the driver class is executed?

```
public interface StackADT {
      public int size();
      public void insertAtHead(int val);
      public void insertAtTail(int val);
      public void insertInMiddle(int val);}
```

```
public class BDriver {
     public static void main(String[] args){
      int[] arr={2, 5, 7};
      B b = new B(arr);
      b.insertAtHead(4);
      b.insertAtTail(6);
      System.out.print(b);
     }
}
```

```
public class B implements StackADT {
      private int[] arr;
      public B(int capacity) {
            arr = new int[capacity];}
      public int size(){
            return arr.length;}
      public void insertAtHead(int val){
            arr[0]=val;}
      public void insertAtTail(int val){
            arr[arr.length-1]=val;}
      public String toString(){
            String output= "";
      for(int i=0;i<arr.length;i++)
            output+=arr[i];}
}
```

a. `2 6 7`
b. `4 5 6`
c. It produces a run-time error
d. It doesn't compile correctly
e. None of the above

Problem IV: Evaluating Java Expressions (**20 minutes**) [20 points]

For each of the following code fragments, what is the value of **x** after the statements are executed? (5 pts)

(1)
```
int x = 0;
int[] A = {0, 1, 2, 3, 4};
for(int i=0; i< A.length-1; i+=2)
{
   x = A[i]; A[i] = ++x; ++A[i];
}
```
Answer: x= **3**

(2)
```
int x = 1;
int [] T = { -1, 1, -2, 2, -3, 3 };
for ( int n = 1; n < T.length; n++)
      x += -1 * n + T[n];
```
Answer: x= **-13**

(3)
```
int [] A = new int[4];
for (int i = 0; i < A.length; i++) {
A[i] = i * 3; int x = A[i]; } int x = 1;
for (int i = 0; i < A.length/2; i++)
x += A[i];
```
Answer: x= **4**

(4)
```
int [] T = new int[5];
int x = 0; int i=0;
for (i = T.length - 1; i >= 1; i-=1 ) {
T[i] = i * i; x++; }
x = T[i]+T[i+1];
```
Answer: x= **1**

(5)
```
String y = "Only the educated remain free!";
String x =""; int i = 0;
Scanner scan = new Scanner(y);
while (scan.hasNext())
{
     x += scan.next().charAt(i);
       i++;
}
```
Answer: x= **“Ohua!”**

(6)
```
ArrayList <Integer> A = new ArrayList <Integer>();
A.add(10); A.add(20); A.add(30); A.add(40); int x = 0;
for(int i =0; i<A.size()-1; i++)
{ x += A.get(i); A.remove(i+1); }
```
Answer: x= **40**

(7)
```
int x = 0; int [] arr = new int[4];
for (int k = arr.length - 1; k >= 0; k -=1 )
{arr[k] = k + 10 + x; x = arr[k];}
```
Answer: x= **46**

For each of the following code fragments, what are the values of the array elements after the statements are executed? (6 pts)

(8)
```
int[] T = {2, 2, 2, 3, 3, 3};
for (int i = 1; i < T.length; i++) {
    T[i] = (T[i-1] + T[i])%i;
}
```

Indexes	0	1	2	3	4	5
Values	**2**	**0**	**0**	**0**	**3**	**1**

(9)
```
int[][] T = {{1, 2, 1}, {2, 3, 2}, {3, 4, 3}};
   for (int i = 0; i < T.length; i++)
       for(int j=0; j<T[i].length; j++)
            T[i][j] = T[j][1] + T[1][i];
```

Indexes	0	1	2
0	**4**	**5**	**6**
1	**8**	**6**	**10**
2	**15**	**16**	**26**

(10)
```
    int[][] T = {{1, -1, 2}, {3, -3, 4}, {4, -4, 5}};
for (int j = 0; j < T.length; j++)
   for(int i=0; i<T[j].length; i++)
       T[i][j] += T[j][i];
```

Indexes	0	1	2
0	**2**	**1**	**8**
1	**2**	**-6**	**4**
2	**6**	**0**	**10**

Problem V: Method definition (**15 minutes**) [10 points]

You are given below the descriptions of 2 static methods called `IndexOfMinNeg` and `MaxPerCol`. Your job is to complete the definition (including the header and body) of each one of these methods as per the provided guidelines.

1. `IndexOfMinNeg` is a method that accepts a 1D array of integers `A`, and then returns the index of the smallest negative value within the elements of the array. The method should return -1 if no negative values are found in the array.

```
public static int IndexOfMinNeg (int[] A)
{


      int iMinNeg = -1;
      int iMin = 0;
      for (int i=0; i < A.length; i++)
      {
         if(A[i] <0 && A[i] < A[iMin])  iMin = i;
      }

      if(A[iMin] < 0)   iMinNeg = iMin;
      return iMinNeg;


}
```

2. `MaxPerCol` is a method that accepts a 2D array of integers `A`, and then returns a 1D array storing the maximum of the group of numbers that are stored in each of the 2D array's columns.

```
public static int[] MaxPerCol(int[][] A)
{

    int NbRows = A.length;
    int NbCols = A[0].length;

    int [] Max = new int [NbCols];

    for(int j=0; j< NbCols; j++)
    {   Max[j] = A[0][j];
        for (int i=1; i < NbRows; i++)
         {  if(A[i][j] > Max[j])  Max[j] = A[i][j];  }
     }

     return Max;

}
```

Problem VI: Coding (**40 minutes**) [40 points]

1. Write a Java program called `ReverseRoundedArray` that reads from the user first the size and then the values of an array `A` of double numbers. Then, the program produces an array of integer numbers `B` with the same size as `A`, and having as values those of `A` rounded to their "floor" numbers, and appearing in reverse order in `B` (**as shown in the sample output below**).

   Sample output:
   Enter size: -5
   Wrong input! Size should be positive! Try again: 5
   Enter integer elements of array A: 2.96 10.4 4.875 3.57 4.03
   Elements of array B: 4 3 4 10 2

```
import java.util.Scanner;

class ReverseArray{


      public static void main (String [] args)
      {

          Scanner scan = new Scanner (System.in);

          System.out.print("Enter size:");
          int size = scan.nextInt();

          while(size <=0)
          {
             System.out.print("Wrong input! Size should be positive! "+
                              " Try again:");
              size = scan.nextInt();
          }

          double [] A = new double[size];

          System.out.print("Enter elements of array A: ");

          for(int i=0; i< A.length; i++)
               A[i] = scan.nextDouble();


          int [] B = new int[size];
          for (int i= 0; i< B.length; i++)
               B[i] = (int)Math.floor(A[A.length-1-i]);

          System.out.print("Elements of array B: ");
          for(int i=0; i< B.length; i++)
              System.out.print(B[i] + "   ");
      }
}
```

2. Write a Java program called `SplitPosNeg` that reads from the user first the size and then the values of an array `A` of integer numbers. Then, the program creates an array of integer numbers `B` with the same size as `A` as follows. `B` should include first the positive or null elements of `A`, followed by the negative elements of `A`.

Sample output:
Enter size of array: 8
Enter elements of array A: -4 25 -7 -19 35 101 49 -1000
Elements of array B: 25 35 101 49 -4 -7 -19 -1000

```
import java.util.Scanner;

public class SplitPosNeg{

   public static void main(String[] args) {

   Scanner scan = new Scanner (System.in);
   System.out.print("Enter size of array:");
   int size = scan.nextInt();

   while(size <=0)
   {
       System.out.print("Wrong input! Size should be positive!"+
                        "Try again:");
       size = scan.nextInt();
   }

   int [] A = new int[size];

   System.out.print("Enter elements of array A: ");

   for(int i=0; i< A.length; i++)
       A[i] = scan.nextInt();

   int [] B = new int[size];
   int j=0;

   for(int i=0; i< A.length; i++)
   {
        if(A[i] >=0) { B[j] = A[i]; j++;}
   }

   for(int i=0; i< A.length; i++)
   {
      if(A[i] <0) { B[j] = A[i]; j++;}
   }

   System.out.print("Elements of array B: ");
   for (int i=0; i<B.length; i++)
        System.out.print(B[i] + "  ");

   }
}
```

3. Write a Java program called `ReadingArray` that reads from the user first the size and then the values of an array `A` of integer numbers. Then, the program will read from the user another array `B` of integers having the same size as `A` and where the program should, when reading the elements of `B`, verify that elements in `B` and `A` occurring at the same index have a difference that is less than or equal to 2. If such a condition is not satisfied for a certain index in `B`, the user should be asked to enter another value for that index (**as illustrated in the sample output below**). The program finally prints out the elements of `B`.

Sample output:

Enter size of array: 4
Enter integer elements of array A: 1 2 3 -4
Now, enter integer elements of array B
Enter element B[0]: 2
Enter element B[1]: 2
Enter element B[2]: 0
Difference between B[2] and A[2] is > 2! Enter element B[2] again: 1
Enter element B[3] = -3
Elements of array B: 2 2 1 -3

```
import java.util.Scanner;

public class ReadingArray{
   public static void main(String[] args) {

       Scanner scan = new Scanner (System.in);
       System.out.print("Enter size of array:");
       int size = scan.nextInt();
       while(size <=0)
       {
           System.out.print("Wrong input! Size should be positive! Try again:");
           size = scan.nextInt();
       }

        int [] A = new int[size];
        System.out.print("Enter integer elements of array A: ");
        for(int i=0; i< A.length; i++)
            A[i] = scan.nextInt();

        int [] B =  new int[size];
        System.out.println("Now enter integer elements of array B");
        for(int i=0; i< B.length; i++)
        {
             System.out.print("Enter element B[" + i + "]: ");
             B[i] = scan.nextInt();
             while(Math.abs(B[i]-A[i])>2)
             {
                  System.out.print("Difference between B[" + i + "] and A[" +
                                   i + "] is > 2! Enter element B[" + i + "] again: ");
                  B[i] = scan.nextInt();
             }

         }

         for(int i=0; i< B.length; i++)
             System.out.print(B[i] + "  ");
   }
}
```

4. Write a Java program that reads from the user an array of integer numbers and a positive integer x, and then counts the number of times x identical numbers occur next to each other in the array.

Sample output:
Enter size of array: 8
Enter integer elements of array: 5 2 2 2 -3 -1 -1 -1
Enter x: 3
The number of occurrences of 3 consecutive identical numbers: 2

```
import java.util.Scanner;

public class ConsecutiveInts{
   public static void main(String[] args) {

       Scanner scan = new Scanner (System.in);
       System.out.print("Enter size of array:");
       int size = scan.nextInt();

       while(size <=0)
       {
           System.out.print("Wrong input! Size should be positive!"+
                            "Try again:");
           size = scan.nextInt();
       }

        int [] A = new int[size];
        System.out.print("Enter elements of array A: ");
        for(int i=0; i< A.length; i++)
             A[i] = scan.nextInt();

         int x;
         do{
             System.out.print("Enter x: ");
             x = scan.nextInt();
         } while (x<=0);

         int count = 0;
         int i=0;
         while(i<= A.length-x)
         {
             boolean consec = true;
             int j=i+1;
             while( consec && (j-i)<x)
             {
                  if (A[i] != A[j]) consec = false;
                  else j++;
              }
            if(consec)  count++;
            i=j;
         }

     System.out.print("The number of occurrences of " + x +
                      " consecutive identical numbers: " + count);
   }
}
```