Name (print): ________________________________________________

• INSTRUCTIONS:

  – Keep your eyes on your own paper and do your best to prevent anyone else from seeing your work.
  – Do NOT communicate with anyone other than the professor/proctor for ANY reason in ANY language in ANY manner.
  – This exam is closed notes, closed books, and no calculator.
  – Turn all mobile devices off and put them away now. You cannot have them on your desk.
  – You cannot use the Internet!
  – Write neatly and clearly indicate your answers. What I cannot read, I will assume to be incorrect.
  – Stop writing when told to do so at the end of the exam. I will take 5 points off your exam if I have to tell you multiple times.
  – Academic misconduct will not be tolerated. Suspected academic misconduct will be immediately referred to the Emory Honor Council. Penalties for misconduct will be a zero on this exam, an F grade in the course, and/or other disciplinary action that may be applied by the Emory Honor Council.

• TIME: This exam has 7 problem. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Emory community. I have also read and understand the requirements and policies outlined above.

Signature: ________________________________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points:</td>
<td>6</td>
<td>20</td>
<td>10</td>
<td>14</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Scope
   (a) (1 point) Give a definition of the scope of a variable (in your own words).

   **Solution:** The scope of a variable is the part of a program where it can be referenced.

   (b) (5 points) For the program below specify the scope of each variable (just specify the range of lines).

   ```java
   public class Scope {
   public static int method(int a) {
      int b = 10;
      return a + b;
   }
   
   public static void main(String[] args) {
      int a = 5;
      {  
         int c = method(a);
         
         for (int i = 0; i < 10; ++i) {
            int c = method(i);
         }
      }
   }
   }
   ```

   parameter variable a (declared at line 2): [2-5]
   variable b (declared at line 3): [3-5]
   parameter variable args (declared at line 7): [7-15]
   variable a (declared at line 8): [8-15]
   variable c (declared at line 10): [10-11]
   variable i (declared at line 12): [12-14]
   variable c (declared at line 13): [13-14]

2. Methods
   (a) (1 point) What is the output of the program below?

   ```java
   public class Problem2a {
      public static void myMethod() {
         System.out.println("World");
      }
   
      public static void main(String[] args) {
         System.out.println("Hello");
      }
   }
   ```
(b) (7 points) What is the output of the program below?

```java
public class Problem2b {
    public static void main(String[] args) {
        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j <= i; ++j) {
                for (int k = 0; k <= j; ++k) {
                    System.out.println(i + " , " + j + " , " + k);
                }
            }
        }
    }
}
```

**Solution:**
0, 0, 0
1, 0, 0
1, 1, 0
1, 1, 1
2, 0, 0
2, 1, 0
2, 1, 1
2, 2, 0
2, 2, 1
2, 2, 2
(c) (12 points) What is the output of the program below?

```java
public class Problem2c {
    public static void strangeMethod(String i, String love, String java) {
        System.out.println(i + " " + love + " " + java);
    }

    public static void main(String[] args) {
        String i = "java";
        String love = "i";
        String java = "love";
        strangeMethod(i, love, java);
        strangeMethod(i, "love", java);
        strangeMethod("i", "love", "java");
    }
}
```

Solution:
java i love
java love love
i love java

3. Methods and pass by value

(a) (2 points) Methods defined in a Java program can be called from different places in the same program or even in other programs. But how does Java know which statement to execute upon returning from the method?

Solution: The address of the instruction to return to is stored on the call stack.

(b) (5 points) Look at the code below. What does it output and why don’t we see the same output in both cases?

```java
public class PassByValue {
    public static void doit(int[] arr, int size) {
        int[] res = new int[size];
        for (int i = 0; i < arr.length; ++i)
            res[i]=arr[i];
        arr = res;
        arr[0] = 50;
    }

    public static void doit(int[] arr) {
        arr[0] = 50;
    }

    public static void main(String[] args) {
        int[] a = {1,2,3,4,5};
        int[] b = {1,2,3,4,5};
```
doit(a, 10);
System.out.println(a[0]);
doit(b);
System.out.println(b[0]);
}
}

**Solution:** 1
50
Java uses pass by value. But in case of arrays and other non-primitive data types the value of a variable is a reference to the memory location storing actual data. When we pass an array to a method reference is passed by value, this means that a new parameter variable is created and this variable stores the same reference as the actual parameter you passed to the method. Thus, if you change this reference inside the method, the value of the actual parameter will not change (that’s why in case of doit method which create a new array of larger size we don’t actually change the value of array a in the main method). However, if we modify elements of array we also modify elements of array references by the actual parameter, because actual parameter and parameter inside the method reference the same array.

(c) (3 points) Below there are 2 methods, that take an array and find its minimum value. Which method is better and why?

```java
public static void getMinimum1(int[] arr) {
    int mn = arr[0];
    for (int i = 1; i < arr.length; ++i) {
        if (mn > arr[i])
            mn = arr[i];
    }
    System.out.println("Minimum is " + mn);
}

public static int getMinimum2(int[] arr) {
    int mn = arr[0];
    for (int i = 1; i < arr.length; ++i) {
        if (mn > arr[i])
            mn = arr[i];
    }
    return mn;
}
```

**Solution:** getMinimum2 is better, because it acts like a black box, taking an array and returning the minimum element. This method can be used in many other programs. You don’t usually want printing to happen as a side-effect of a method call. Imaging that you are writing a program and using Math.min() method, it won’t be very good if the method prints something every time you
call it. Also, since the first method doesn’t return anything you cannot use the minimum value calculated by the method, which also decreases the value of the method.
4. (14 points) What is the output of the following program?

```java
public class Output {
    public static int a = 10;

    public static int foo() {
        System.out.println(a);
        return a;
    }

    public static int foo(int a) {
        System.out.println(a);
        return a;
    }

    public static int foo(double b) {
        if (b > 0) {
            int a = (int)b;
            System.out.println(a);
            a = 100;
        }
        return a;
    }

    public static void foo(int a, int b) {
        Output.a = a + b;
    }

    public static void main(String[] args) {
        {
            int a = 1;
            int b = foo();
            System.out.println(foo(a));
            foo(a, b);
            System.out.println(Output.a);
            System.out.println(foo(10/5));
        }
        {
            double a = 1.0;
            System.out.println(foo(a));
        }
    }
}
```

Solution:
10
1
1
11
2
5. Methods

(a) (7 points) Write a method, that takes an array and indexes a and b and calculates
the sum of array elements starting at element with index a and up to including
element with index b (make your method return 0 in case there is an error with the
indexes - not in the range of elements or a > b).

```
public static int sumElements(int[] arr, int a, int b) {
    Solution:
    if (a > b || a < 0 || b >= arr.length)
        return 0;
    int sum = 0;
    for (int i=a; i<=b; ++i)
        sum += arr[i];
    return sum;
}
```

(b) (3 points) Assume the method you wrote in part(a) works as specified. Using this
method, write an overloaded method that takes an array as an input parameter and
calculates the sum of all the elements in the array.

```
Solution:
public static int sumElements(int[] arr) {
    return sumElements(arr, 0, arr.length-1);
}
```

6. Imagine that government of one small country decided to track traffic situation on one
particular road segment. For a day (24 hours) the total number of cars that enter and
exit this road segment was recorded (assume this is a one-way road and cars do not stop
driving once they enter the road segment). To store this data 2 arrays of integer values
were used. One array stores how many cars entered the segment in 5 mins and the other
stores how many cars left the segment. These numbers do not have to be equal as cars
may travel at different speeds through the road segment. The data for the first 5 minute
period is stored in element 0, next 5 mins - in the element 1, etc for 24 hours.

You were asked to do some analytics on this data. Write methods, that calculate the
following:
(a) (5 points) How many cars were there on a road segment at the end of each 5 minute period (we assume that at the beginning there were no cars).

```java
public static int[] getNumberOfCars(int[] carsEntered, int[] carsLeft) {
    int[] numberOfCars = new int[carsEntered.length];
    numberOfCars[0] = carsEntered[0] - carsLeft[0];
    for (int i = 1; i < carsEntered.length; ++i) {
        numberOfCars[i] = numberOfCars[i - 1] + carsEntered[i] - carsLeft[i];
    }
    return numberOfCars;
}
```

Solution:

```java
class
```

(b) (7 points) Write a method to calculate how long (in hours (not 5 minute periods)) there was a traffic jam on this road segment. A situation is called a traffic jam if there are more than 50 cars on the segment at the end of a 5 minute period (you can use method defined in the previous problem). You may assume that the method you wrote in part(a) works as specified.

```java
public static double getTrafficJamDuration(int[] carsEntered, int[] carsLeft) {
    double jamDuration = 0;
    int[] cars = getNumberOfCars(carsEntered, carsLeft);
    for (int i = 0; i < cars.length; ++i) {
        if (cars[i] > 50)
            ++jamDuration;
    }
    return jamDuration / 12.0; // to get hours
}
```

(c) (8 points) Write a method that takes an array carsEntered and returns another array where each element corresponds to 1 hour period instead of 5 minute period. In other words, the result of the method should be an array of 24 values, where each element is the number of cars entered the segment during the corresponding hour.

```java
public static int[] aggregateData(int[] carsEntered) {
    int[] data = new int[24];
    for (int i = 0; i < carsEntered.length; ++i) {
        int hour = i / 5;
        data[hour] += carsEntered[i];
    }
    return data;
}
```

Solution:
```java
int[] res = new int[24];
for (int i = 0; i < carsEntered.length; ++i) {
    res[i] += carsEntered[i / 12]; // Each 12 consecutive element goes to the same position
}
return res;
```
7. (20 points) Write a program, that asks user to enter the size of the table (n and m) and then reads the table of double values of size n by m. Then your program needs to find the column with the largest sum of elements and print out this column (as a column, each element on a new line). Using methods can make your life better, at least in this problem. Declare a method that calculates the sum of elements in a column, then use this method to find a column with the largest sum and print it.

Solution:

```java
import java.util.Scanner;

public class LargestColumn {
    public static double sumColumn(double[][] table, int columnIndex) {
        double sum = 0;
        // Sum all elements in the columnIndex column by going over all rows.
        for (int rowIndex = 0; rowIndex < table.length; ++rowIndex) {
            sum += table[rowIndex][columnIndex];
        }
        return sum;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Please enter the size (number of rows and columns) of the table: ");
        int rowCount = input.nextInt();
        int columnCount = input.nextInt();
        // Create a table
double[][] table = new double[rowCount][columnCount];
        System.out.println("Please enter the values of the table:");

        // Read input using nested loops for all rows and all columns.
        for (int rowIndex = 0; rowIndex < rowCount; ++rowIndex) {
            for (int columnIndex = 0; columnIndex < columnCount; ++columnIndex) {
                table[rowIndex][columnIndex] = input.nextDouble();
            }
        }

        // This variable will store the current maximum sum of element in one column. Initialize with sum of column 0.
double maxColumnSum = sumColumn(table, 0);

        // This variable will store index of the best column.
        int maxColumnIndex = 0;

        // Try all columns.
        for (int columnIndex = 1; columnIndex < columnCount; ++columnIndex) {
            double currentColumnSum = sumColumn(table, columnIndex);
            if (currentColumnSum > maxColumnSum) {
                maxColumnSum = currentColumnSum;
                maxColumnIndex = columnIndex;
            }
        }

        System.out.println("The column with the largest sum is: ");
        for (int rowIndex = 0; rowIndex < rowCount; ++rowIndex) {
            System.out.println(table[rowIndex][maxColumnIndex]);
        }
    }
}
```
columnIndex) {
    // Calculate the sum of elements in the column.
    double sum = sumColumn(table, columnIndex);
    // Check if it is larger than the current best sum.
    if (sum > maxColumnSum) {
        maxColumnSum = sum;
        maxColumnIndex = columnIndex;
    }
}

// Print the column with index maxColumnIndex.
System.out.println("Column with the largest sum is:");
for (int rowIndex = 0; rowIndex < rowCount; ++rowIndex) {
    System.out.println(table[rowIndex][maxColumnIndex]);
}
}