Problem 1: Basic Java and Pseudocode (30 points)

1. **(4 points)** Evaluate the following expression and select the type and value of the result

(int) 77.77

a. int, 77
b. double, 77.77
c. double, 77.0
d. long, 77
e. int, 78

2. **(4 points)** Evaluate the following expression and select the type and value of the result

(6 != 6) || (2 == 2) && (3 == '3')

a. boolean, true
b. boolean, false
c. int, true
d. Compile Error
e. Runtime Error

3. (**4 points)** Evaluate the following expression and select the type and value of the result

Boolean.parseBoolean("True") || (7 != (77/11))

a. boolean, true
b. boolean, false
c. int, true
d. Compile error

4. **(7 points)** Examine the following pseudocode algorithm. What is the output and what is the total number of operations that are executed by this code?

```
Input : none
Output : ???????
Pre-condition : none

SET count TO 0
SET end TO 10
WHILE count < end
   DISPLAY count
   ADD 1 TO count
ENDWHILE
```

Answer:
Output: 0 1 2 3 4 5 6 7 8 9 // 3 points
Number of operations : 2 (init ops) + 3N + 1 (failed while comp) - 2 + 30 + 1 = 33 // 4 points

5. **(6 points)** Examine the following snippet of code:

```
int a = 4, b = 6, c = 0;
for (int i = 0; i < a; i++)
{
    c += a;
    for (int j = 0; j < b; j++) {
       c += b;
    }
}
StdOut.println(c);
```

What is the output?
Answer: 160

6. **(5 points)** Given the code below add a line that prints the Numbers array to the screen in reverse.
Place you code where the XXX's are.

```
public class TestArrays
{
   public static void main(String[] args)
   {
      // Populate the array
      int numbers[] = {1, 2, 3, 4, 5, 6, 7};

      // Print from index 0 to N
      for (int i = 0; i < numbers.length; i++) {
         // Print the array item at index i
         StdOut.print(numbers[i]);
      }
      StdOut.print('\n');

      // Print the array in reverse
      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

   }
}
```

Answer : for (int i = (Numbers.length-1); i >= 0; i--) StdOut.print(Numbers[i]);

Problem 2: Functions (30 points)

1. **(5 points)** What does the following program do?

```
public static void negate (int value) {
    value = -value;
}

public static void main (String[] args) {
    int a = 43;
    System.out.println(a);
    negate(a);
    System.out.println(a);
}
```

a) Prints the integer 43 twice
b) Prints the integers 43 and then -43
c) Prints the integer -43 twice
d) None of the above

2. **(15 points)** Write a static method `nearestInt()` that has one double parameter value and returns the nearest integer. Do not use any Math library function, instead use casting.

   If the digit in the tenths place is less than 5, then **round down**, which means the units digit remains the same; if the digit in the tenths place is 5 or greater, then round up, which means you should increase the unit digit by one.

   Examples:

```
nearestInt(2.5  ) returns 3
nearestInt(2.49 ) returns 2
nearestInt(-2.5 ) returns -3
nearestInt(-2.49) returns -2
```

   Solution:

```
public static int nearestInt (double n) {

    int nint = (int) n; // cast

    if ( n > 0 ) {
        // n is positive
        if ( (n - nint) >= 0.5 )
            nint += 1;
    } else {
        // n is negative
        if ( (nint - n) >= 0.5 ) {
            nint -= 1;
        }
    }

    return nint;
}
```

```
// 5 points for method signature, no partial credit
// 2 points for casting
// 3 points for handling positive numbers correctly
// 3 points for handling negative numbers correctly
// 2 point for the returning an integer value
```

3. **(10 points)** Complete the method `anyTrue()` that takes an array of `boolean` values as its argument and returns `true` if any of the elements in the array is `true`, and `false` otherwise.

   For example:
   `anyTrue ([true,true,true,true])` returns `true`
   `anyTrue ([false,false,false,true])` returns `true`
   `anyTrue ([false,false,false,false])` returns `false`

```
public static boolean anyTrue (boolean[] a) {

}
```

Solution:

```
public static boolean anyTrue (boolean[] a) {

    for ( int i = 0; i < a.length; i++ ) {
        if ( a[i] == true ) return true;
    }
    // all items are false
    return false;
}
```

```
// 5 points for iterating over the array
// 3 points for returning true if the array contains a true value
// 2 points for returning false if array contains no true value
```

Problem 3: Recursion (35 points)

Questions 1 - 3 refer to the following three code segments (labeled A, B, and C).

A.

```
public static void myMethod(int n){
    if (n>0){
        myMethod(n-1);
    }
    System.out.println(n);
}
```

B.

```
public static void myMethod(int n){
    if (n>0){
        myMethod(n-1);
        System.out.println(n);
    }
}
```

C.

```
public static void myMethod(int n){
    if (n>0){
        myMethod(n-1);
    } else{
        System.out.println(n);
    }
}
```

1. Which statement about methods A, B, and C above is true if the myMethod is called with a positive int parameter? Answer: b (5 points)
    a. All three methods produce the same output.
    b. All three methods produce different outputs.
    c. Methods A and B produce the same output that is different from Method C.
    d. Methods A and C produce the same output that is different from Method B.
    e. Methods B and C produce the same output that is different from Method A.
2. If the statement
    myMethod(5);
    were executed, what is the output produced by code segment B?
    Answer (ok to put on one line): (6 points)
    1
    2
    3
    4
    5
3. For this question, **consider Code Segment C only**. If the statement
    myMethod(5);
    were executed, what is the total number of times myMethod is called (including the original call)?
    Answer: 6 (6 points - Give 3 pts for answer of 5)

4. Given the recursive method surprise below:

```
public static String surprise(String s){
    if (s.length() <= 1){
        return s;
    }
    return  s.charAt(s.length() - 1) + surprise(s.substring(0,s.length() - 2));
}
```

What is the printed when the following instruction is executed?

```
System.out.println(surprise("mathematics"));
```

Answer: siaetm (6 points)

5. Consider the recursive method magic below.

```
public static boolean magic(int[] a, int size){
    if (size == 1){
        return true;
    }
    return (a[size-1] > a[size - 2] && magic(a, size - 1) );
}
```

Suppose the following code segment is executed. Assume that args.length > 0.

```
int[] arr = new int[args.length];
for(int i = 0; i < args.length; i++){
    arr[i] = Integer.parseInt(args[i]);
}
System.out.println(magic(arr, arr.length));
```

a. Give a set of input values that would result in **true** being printed. Answer: any set of strictly increasing values. (3 points)
b. Give a set of input values that would result in **false** being printed. Answer: any set of NON-strictly increasing values (3 points)
c. Is there any set of values that would cause an endless loop? Answer: no (3 point)
d. Describe the intended purpose of the method magic? Returns true if the values in an array are in strictly increasing order; returns false otherwise. (3 points)

Problem 4: OOP Using Data Types (35 points)

Suppose we have a String in the form “s1;s2;s3;..;sn;” where s1 to sn all have the form “name,id,gpa”. Here, name, id and gpa separately represent a student’s name, id and gpa.
And the three pieces of information are separated by the character ‘,’ and there is at least one character there for each kind of information. Note that the string always ends with one ‘;’.

The following code is a class defined to represent a student:

```
public class Student {
   private String name;
   private int id;
   private double gpa;

   public Student(String name, int id, double gpa) {
      this.name = name;
      this.id = id;
      this.gpa = gpa;
   }

   public String toString(){
      return "Name: " + this.name + "\nID: " + this.id + "\nGPA: " + this.gpa + "\n";
   }
}
```

Write a method that takes such a string and an integer n as inputs, where n represents the number of ‘;’ in the string and returns a 1-D array containing n Student objects where each object is constructed using the information of si which is in the form of “name,id,gpa”.

For example, if the parameters are {"Joy,101,3.45;Emily,102,3.56;Nancy,103,3.21;Jimmy,104,4.0;"} and 4, then the output would be a 1-D array whose length is 4 and has the following four Student objects:

```
Name: Joy
ID: 101
GPA: 3.45

Name: Emily
ID: 102
GPA: 3.56

Name: Nancy
ID: 103
GPA: 3.21

Name: Jimmy
ID: 104
GPA: 4.0
```

Use the provided String library methods to operate these string values. Here is the documentation for String:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

The method signature is given below:
public static Student[] convert(String str, int n)

Sol:

```
public static Student[] convert(String str, int n){
    Student[] array = new Student[n];
    int i = 0;
    while(str.contains(";")){
        int index1 = str.indexOf(",");
        String name = str.substring(0, index1);

        str = str.substring(index1+1);
        int index2 = str.indexOf(",");
        String idStr = str.substring(0, index2);
        int id = Integer.parseInt(idStr);

        str = str.substring(index2+1);
        int index3 = str.indexOf(";");
        String gpaStr = str.substring(0, index3);
        double gpa = Double.parseDouble(gpaStr);

        array[i] = new Student(name, id, gpa);
        i++;

        str = str.substring(index3+1);
    }
    return array;
}
```

Grading:
3 pts for creating a 1-D array with n.
5 pts for a loop to handle each student in the parameter str.
3 pts for separating different students, the sample solution uses indexOf, if the students use charAt to check each character, it is also ok. There might be some other solutions too. But must use methods in the String class. Otherwise, no points.
3 pts for getting the name, the sample solution uses substring. But must use methods in the String class. Otherwise, no points.
3 pts for getting the id, the sample solution uses substring. But must use methods in the String class. Otherwise, no points.
3 pts for using Integer.parseInt to convert String to int.
3 pts for getting the gpa, the sample solution uses substring. But must use methods in the String class. Otherwise, no points.
3 pts for using Double.parseDouble to convert String to double.
4 pts for calling the constructor to create a Student object.
2 pt for setting array element a[i]
3 pts for returning a 1-d array value.