Problem 1: Algorithm Implementation and Basic Java (30 points)

a) **(10 points)** The following code is supposed to display the average of five command-line integer arguments, but it doesn't display the correct value. What is the problem with this code? How to fix it in order for it to display the average of the five command line integer arguments?

```java
public static void main (String[] args) {

    int sum = Integer.parseInt(args[0]);
    sum += Integer.parseInt(args[1]);
    sum += Integer.parseInt(args[2]);
    sum += Integer.parseInt(args[3]);
    sum += Integer.parseInt(args[4]);

    double average = sum/5;
    System.out.println(average);
}
```

The problem is that the result of sum/5 is an integer. **[5 points]**
It needs to be converted into a double by (either solution is correct): **[5 points]**

- double average = (double) sum/5;
- double average = (sum*1.0)/5;

b) **(5 points)** The following code command-line inputs are a weekday (Monday through Sunday), the weekday temperature in Fahrenheit, and the conversion scale. It displays the converted temperature (Kelvin or Celsius) for the weekday.

```java
public class Convert {
    Run | Debug
    public static void main (String[] args) {

        String weekDay      = args[0];
        double temperatureF = Double.parseDouble(args[1]);
        int    scale        = Integer.parseInt(args[2]);  // 1 for Kelvin, 2 for Celsius

        if ( scale == 1 ) {

            double result = (temperatureF - 32) * 5/9 + 273.15;
            System.out.println(weekDay + " temperature in Kelvin is " + result);

        } else if ( scale == 2 ) {

            double result = (temperatureF - 32) * 5/9;
            System.out.println(weekDay + " temperature in Celsius is " + result);

        } else {
            System.out.println("Incorrect scale: [1]-Kelvin, [2]-Celsius");
        }
    }
}
```

Which of the following statements executes the program properly displaying the converted temperature, choose all that apply?

a. java Convert Monday 78 2 (correct)
b. java Convert 89 Monday 1
c. java Convert Thursday 89.4 1 (correct)
d. java Convert Sunday 23.6 2 (correct)

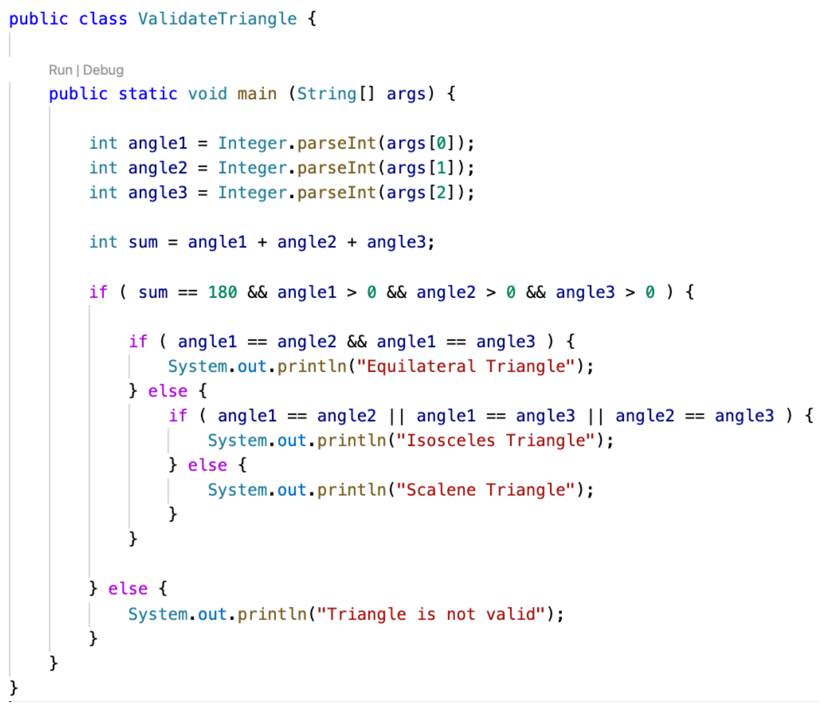
e. java Convert Tuesday 32 1 (correct)
f. All of the above

c) **(15 points)** Translate the following pseudocode into a Java class called `ValidateTriangle`.

```
READ angle1
READ angle2
READ angle3

COMPUTE sum AS angle1 + angle2 + angle3

IF  sum equals 180 AND angle1 > 0 AND angle2 > 0 AND angle3 > 0 THEN
  IF angle1 equals angle2 AND angle1 equals angle3 THEN
    DISPLAY Equilateral triangle
  ELSE
      IF angle1 equals angle2 OR angle1 equals angle3 OR angle2 equals angle3 THEN
        DISPLAY Isosceles triangle
      ELSE
        DISPLAY Scalene triangle
      ENDIF
   ENDIF
ELSE
  DISPLAY Triangle is not valid
ENDIF
```



```java
public class ValidateTriangle {

    public static void main (String[] args) {

        int angle1 = Integer.parseInt(args[0]);
        int angle2 = Integer.parseInt(args[1]);
        int angle3 = Integer.parseInt(args[2]);

        int sum = angle1 + angle2 + angle3;

        if ( sum == 180 && angle1 > 0 && angle2 > 0 && angle3 > 0 ) {

            if ( angle1 == angle2 && angle1 == angle3 ) {
                System.out.println("Equilateral Triangle");
            } else {
                if ( angle1 == angle2 || angle1 == angle3 || angle2 == angle3 ) {
                    System.out.println("Isosceles Triangle");
                } else {
                    System.out.println("Scalene Triangle");
                }
            }

        } else {
            System.out.println("Triangle is not valid");
        }
    }
}
```

6 points: 2 points each Integer.parseInt()
2 points: correct first compare
3 points: correct second compare
3 points: correct third compare
1 point: display not valid

Problem 2: Loops (30 points)

a) (**8 points**) The following code is intended to continuously get user integer input and print a message based on the values input. Specifically, if the integer is 1, print “create a file.” If the integer is 2, print “open a file.” If the integer is 3, print “close a file.” If the integer is 4, print “delete a file.” The program will end if the user inputs any other integer values. The program does not work as intended. What code do you need to add and where do you need to add this code to fix the program?

In order to fix it, we need add this code else break; in or after line 15

```
1  public class Loop1
2  {
3    public static main (String[] args)
4    {
5      while (true)
6      {
7        int op = StdIn.readInt();
8        if (op == 1)
9           StdOut.println(“create a file.”);
10       else if (op == 2)
11          StdOut.println(“open a file.”);
12       else if (op == 3)
13          StdOut.println(“close a file.”);
14       else if (op == 4)
15          StdOut.println(“delete a file.”);
16     }
17   }
18 }
```

Sol: 4 pts for each

b) (**12 points**) Write a snippet among the integers from 1 to 99, print all integers that are divisible by three. The printed integers should be separated in different lines and each line except the last one has 5 integers. Use a for loop for this question.

Sol:

```
        int count = 0;             //1 pt for the variable used to count
        for(int i=1; i<100; i++){ // 2 pt for initilization, 2 pts for condition, 2 pts for
updation
            if(i % 3 == 0){        // 1 pts for condition checking
                System.out.print(i + " "); // 1 pt for print
                count++;                // 1 pt
            }
            if(count % 5 ==0)           //1 pts for condition checking
                System.out.println();   // 1 pt for println
        }
```

c) (**10 points**) There is a bug in the following code. The error line is line __17 ___. Without modifying the error line, we can fix the bug in this way: ___move the declaration for j before while loop and give it an initial value like 1____.

```
1  public class Loop2
2  {
3    public static main (String[] args)
4    {
5      int n = Integer.parseInt(args[0]);
6      int i = 1;
7      while ( i < n ) {
8         for ( int j = 1; j < n; j++ ) {
9            if ( j % i == 0 )
10              System.out.print(j + “ “);
11           else
12              System.out.print(0 + “ “);
13        }
14        System.out.println();
15        i++;
16     }
17    System.out.println(“There are ” + i + “ rows and ”+ j + “columns”);
18   }
19 }
```

Sol: 4 pts for line 17.
6 pts for the declaration and initialization.
If only mention declaration without initialization, 3 pts.

Problem 3: Arrays (40 points)

a) **(12 points)** For each of the code segments below, indicate if there is a compile-time error, a runtime error, or no error. If there is an error, BRIEFLY describe what the error is and where or why it occurs.

12 points: each box = 1 pt.

Code segment	Compile time (C) or Runtime error (R) or No Error?	Error Description
`int[ ] a = new int[];`	*C*	*No size given*

`int[ ] a = {1, 2, 3, 4, 5};`
`for(int i = 0; i < a.length; i++){`
`   a[i+1] = a[i];`
`}`
	R	*ArrayIndexOutOfBounds-when i= a.length – 1, a[i+1] is out of bounds*

`int [ ][ ] arr = [1,2,3],[4,5,6];`	C	Need { } s `{{1,2,3},{4,5,6}}` or *`int[] arr = {1,2,3,4,5,6}`*

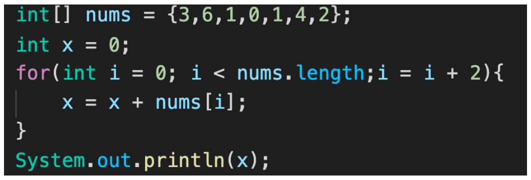
`int[ ] a = new int[10];`
`for(int i = -5; i < 5; i++){`
`   a[i] = i;`
`}`
	R	*ArrayIndexOutOfBounds-index – 5 out of bounds*

`int [ ] [ ] arr = {{1,2,3},{4,5,6}};`
`int[ ] a = new int[arr.length];`
`for (int i = 0; i < arr.length; i++){`
`    for(int j = 0; j < arr[i].length; j++){`
`           a[i] = arr[i][j];`
`    }`
`}`
	No error	

`int[ ] a = {1, 2, 3, 4, 5};`
`for(int i = a.length; i >= 0; i--){`
`    System.out.println(a[i]);`
`}`
	R	*ArrayIndexOutOfBounds-index a[a.length] out of bounds*

b) **(18 points)** Give the output for each of the following code segments.

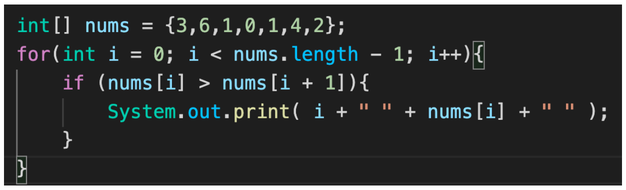
a.



```
int[] nums = {3,6,1,0,1,4,2};
int x = 0;
for(int i = 0; i < nums.length;i = i + 2){
    x = x + nums[i];
}
System.out.println(x);
```

a. Output
7
6 points, no partial credit

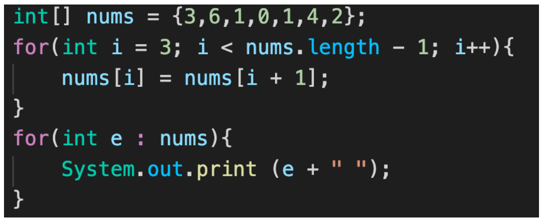
b.



```
int[] nums = {3,6,1,0,1,4,2};
for(int i = 0; i < nums.length - 1; i++){
    if (nums[i] > nums[i + 1]){
        System.out.print( i + " " + nums[i] + " " );
    }
}
```

b. Output:
1 6 2 1 5 4
3 pts for indices (blue numbers)
3 pts for array values (red numbers)

c.



```java
int[] nums = {3,6,1,0,1,4,2};
for(int i = 3; i < nums.length - 1; i++){
    nums[i] = nums[i + 1];
}
for(int e : nums){
    System.out.print (e + " ");
}
```

c. Output:
3 6 1 1 4 2 2
3 pt for 3 values
3 pts for last 4 values

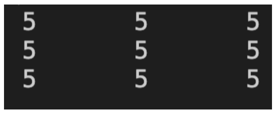
c) **(10 points)** Write a segment of code

A 2D array of `int`s, `nums`, is declared and initialized to contain all 5s. `nums` has the same number of rows and columns. Write a segment of code that will modify the contents of the 2D array, `nums`, resulting with:

- The diagonal of `nums` being unchanged (contains 5s)
- The elements in `nums` that are BELOW the diagonal contain 0s
- The elements in `nums` ABOVE the diagonal contain 1s.

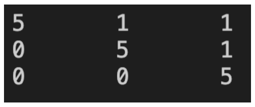
Examples:

Original 2D array

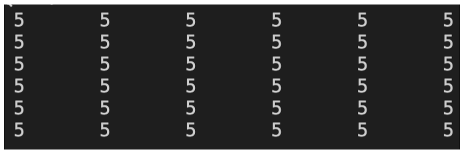


```
5          5          5
5          5          5
5          5          5
```

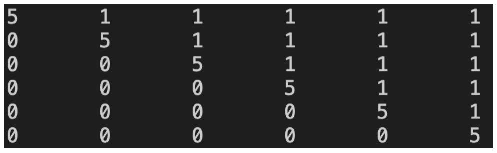
Resulting 2D array



```
5          1          1
0          5          1
0          0          5
```

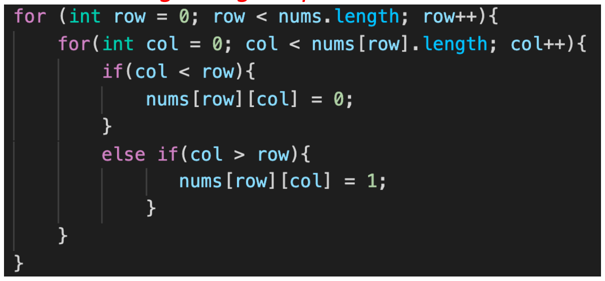


```
5       5       5       5       5       5
5       5       5       5       5       5
5       5       5       5       5       5
5       5       5       5       5       5
5       5       5       5       5       5
5       5       5       5       5       5
```



```
5        1        1        1        1        1
0        5        1        1        1        1
0        0        5        1        1        1
0        0        0        5        1        1
0        0        0        0        5        1
0        0        0        0        0        5
```

Solution and grading: 10 points



```java
for (int row = 0; row < nums.length; row++){
    for(int col = 0; col < nums[row].length; col++){
        if(col < row){
            nums[row][col] = 0;
        }
        else if(col > row){
            nums[row][col] = 1;
        }
    }
}
```

3 parts:
diagonal has 5s : 4 points
col > row has 1s: 3 points
col < row has 0s: 3 points

Problem 4: Input and Output (30 points)

a) **(10 points)** Write a Java program called `SumAve` that takes a variable number of doubles from the command line and displays their sum and their average. For example:

```
java SumAve 7 3 4        will display 14  4.6
java SumAve 11 8 5 20    will display 44  11
```

Solution:

```
public class SumAve
 {
    public static void main (String[] args)
    {
      // take doubles from the command line, sum them and print the average
      // set up variables
      double amt = 0.0;  // 2 points, double summing variable
      int    cnt = 0;

      for (int i = 0; i < args.length; i++) // 5 points, looping over args
      {
          amt = amt + Double.parseDouble(args[i]); // 2 points, read double
          cnt++;
      }

      System.out.println("The sum is: " + amt + " and the average was: " +
(amt/cnt)); // 1 point, displays sum and average (any format)

    }
}
```

b) **(5 points)** The following code is supposed to sum two integer numbers but it does not work as intended. What is wrong with this code?

```
public class FindBug
 {
   public static void main(String[] args)
   {
     // Add two numbers
     int a = Integer.parseDouble(args[0]);
     int b = Integer.parseDouble(args[1]);

     System.out.println("The sum is : " + (a+b));
   }
 }
```

Solution : Class Integer does not have method parseDouble. (No partial credit)

3. **(15 points)** What is the output?

Given the input File numin.dat:

```
 1 2 3 4 5 6 7 8 9 10
   11 12 13 14 15 16 17 18 19 20
   21 22 23 24 25 26 27 28 29 30
   31 32 33 34 55
```

and the Java Code InputArray.java:

```
public class InputArray
 {
   public static void main(String[] args)
   {
    // read data into an array of integers

    int[] intArray = StdIn.readAllInts();

    for (int i = 0; i < intArray.length; i++)
    {
       if (intArray[i] % 11 == 0)
         System.out.print("Check ");
    }

    StdOut.println(intArray[args.length]);
   }
 }
```

We compile to bytecode (javac InputArray.java) and run using redirection:

```
<Linux/Mac> : java InputArray < numin.dat
<Windows> : cat numin.dat | java InputArray
```

What is the output of the program?

Solution:

check check check check 1 <--- 11, 22, 33 and the last number is 55.
1 is intArray[0], there are no arguments
so args.length is 0.

3 points for each check printed
3 points for printing 1