CSE 20

SAMPLE FINAL
Time: 180 minutes

Name
ANSWERS

The following precedence table is provided for your use:

Precedence of Operators
( )
- (unary), !, ++, --
*, /, %
+ , - (binary)
<, <=, >, >=
= =, !=
&&
||
=, +=, -=, *=, /=, %=

Otherwise left to right

Total Possible Points =

1. Match the following statements (i-x) to the outputs (A-J) they produce. **Note: that there are no duplicate answers** ('G' has the value 71)

```
int i=3, j=2;
double d=5.0;
char c = 'G'; // G is 71
```

i. System.*out*.println("OUTPUT is " + ((**int**)c + j)); _____g_____

ii.System.*out*.println("OUTPUT is " + ((**char**)c - j)); _____e_____

iii.System.*out*.println("OUTPUT is " + (**char**)(c - j)); _____b_____

iv.System.*out*.println("OUTPUT is " + (**char**)c + j); _____c_____

v. System.*out*.println("OUTPUT is " + (**char**)(c - d)); _____a_____

vi.System.*out*.println("OUTPUT is " + ((**char**)c - d)); _____d_____

vii.System.*out*.println("OUTPUT is " + (**double**)c + j); _____f_____

viii.System.*out*.println("OUTPUT is " + (**double**)c / j * d); _____j_____

ix.System.*out*.println("OUTPUT is " + c / (**char**)i * d); _____h_____

x.System.*out*.println("OUTPUT is " + c / j * (**int**)d); _____i_____

a) OUTPUT is B

b) OUTPUT is E

c) OUTPUT is G2

d) OUTPUT is 66.0

e) OUTPUT is 69

f) OUTPUT is 71.02

g) OUTPUT is 73

h) OUTPUT is 115.0

i) OUTPUT is 175

j) OUTPUT is 177.5

2. (a) In the following Java code, there are 4 syntax errors and 3 logic errors. Identify each error and explain how to correct it. For example, if a statement was missing a semicolon, circle where the semicolon would be and label the circle "missing ;" or something similar.

```
public class TestIf {
   public static void main(String[] args) {
      Int i = 1;  // int S1

      if (i <= 0); { // extra ; S2 // < L1
            System.out.println(i + " is negative");
      } else if (i == 0) {
            System.out.println("i" + " is zero"); // i L2
      } else if (i >= 0) // > L3
            System.out.println(i + " is positive");
      else
            System.out.println("Should never get here!");

      System.out.println("Done");

   } // left out
}
```

(b) State the 4 errors in the following code. Assume no additional declarations. Circle the errors and describe then with words. For example, if a ';' was missing at the end of a statement, you would circle where the ';' should be and write "missing ;". Some errors here are syntax errors, while some are semantic (logic) errors. The code is SUPPOSED to allocate an array, and then fill each location with the square of the index number.

```
int MAX = 20;
short i = MAX; // can't assign int to a short

int[] values = int[i]; // new

for (i=1; i <= MAX; i--){ // i < MAX; i++
      int[MAX] = i*i; // values[i]
}
```

3. The following code is supposed to write out the numbers from 5 to 1. It has two flaws.

```
public class Loop {

      public static void main(String[] args) {
            int i;

            for(i=10; i>=0; i--)
                  if (i / 2 == 0)
                        System.out.print(i/2 + ",");

      } // end of main

} // end of Loop
```

a) What is the output of the code as it is written?

0,0,

b) Correct the code so that it works as intended. Write the corrected version of the main function in the box below. There is more than one way to fix the problems.

```
            for(i = 10; i > 1; i--)
                  if (i % 2 == 0)
                        System.out.print(i/2 + ",");
```

4. For each value of `n` below, give the output from the execution of the following program segment

a. Consider While loop

```
int i = 1, n;
n = keyboard.nextInt();

while ( i < n){
      System.out.print(i+" ");
      i *= i;
 }
System.out.println("END.");
```

i. `n = 4` 1 1 1 1 1 1 1 1 …

ii. `n = -6` END.

b. Consider Do-While loop

```
int i = 1, n;
n = keyboard.nextInt();
do {
      System.out.print(i+" ");
      i *= i;
} while ( i < n);

System.out.println("END.");
```

i. `n = 4` 1 1 1 1 1 1 1 1 …

ii. `n = -6` 1 END.

c. Consider the If statements

```
int i = 3, n;
n = keyboard.nextInt();
if (i == n)
      System.out.println( "red" );
else
      System.out.println( "green" );
if (i < n)
      System.out.println( "blue" );
else
      System.out.println( "purple" );
```

i. `n = 3`
red
purple

ii. `n = 99`
green
blue

5. Consider the following code: Write code that does the same as the following but uses a `while` loop or a `do-while` loop instead of a `for` loop.

```
int i;
int sum = 0;
for (i = 10; i >= 0; i--) {
   if ( i % 2 == 0)
      sum = sum + i*i;
}

System.out.println(sum);
```

   a. Write code that does the same using a While loop

```
int i = 10;
int sum = 0;
while (i >= 0) {
   if ( i % 2 == 0)
      sum = sum + i*i;
   i--;
}

System.out.println(sum);
```

   b. Write code that does the same using a Do-While loop

```
int i = 10;
int sum = 0;
do {
   if ( i % 2 == 0)
      sum = sum + i*i;
   i--;
} while (i >= 0);

System.out.println(sum);
```

6. a) Consider the code below on the left hand side. The right hand size represents the array created. Fill-in the values of each entry after the code is executed. The first two lines are filled in as an example.

int [] n = new int[6];	0	0	0	0	0	0
int i = 3; n[0] = 5;	5	0	0	0	0	0
n[i] += 5;	5	0	0	5	0	0
n[2+i] = n[0] + 1;	5	0	0	5	0	6
n[4-i] += 1;	5	1	0	5	0	6
n[1] = n[0] + n[1]+ n[2] + n[3] + n[4];	5	11	0	5	0	6
n[1+2] = 1+i;	5	11	0	4	0	6
n[n.length-1] = n[0];	5	11	0	4	0	5
n[n.length-i] = n.length-2;	5	11	0	4	0	5
n[i*2-1] = n[2] * n[2];	5	11	0	4	0	0

b) Consider the code below on the left hand side. The right hand size is the output. If the code does not output anything then leave it blank or write BLANK. First two answers are given as an example.

int i = 0;	BLANK
System.out.println(i);	0
System.out.println(i+2);	2
i *= 2;	BLANK
System.out.println("Num is" + i);	Num is 0
System.out.println(i++);	0
i += 2.0;	BLANK
System.out.println(++i);	4
System.out.println(i % 2);	0
System.out.println(i += 4);	8
System.out.println(i = i * 4);	32
System.out.println (i / 3.0);	10.667

7. a) Suppose a program reads a positive integer from the keyboard into the integer variable `array_length`. Then it will allocate space for an array of `array_length` floats. Use any of the following variables declared already in the rest of the program. You should not use any additional variables.

```
int array_length;
Scanner input = new Scanner(System.in);

double[] d_arr;
int[] i_arr;
float[] f_arr;

// Start here
System.out.print("Enter the max ");
array_length = input.nextInt();
f_arr = new float[array_length];
```

b) Declare and allocate an array of 1000 integers. Write a loop which fills the array locations with their ***reverse*** indices. For example, slot 0 of the array should hold the value 999, and slot 998 of the array should hold the value 1.

```
int [] arr = new int[1000];

for (int i = 0; i < arr.length; i++)
    arr[i] = arr.length – i - 1;
```

c) Assume you have an array of doubles which has already been filled with data read in from the user. Assume this array is called `scores`. We are not sure how long the array is, but we know it is completely full of data. Write code which calculates the average of the **POSITIVE** values in the `scores` array. If there are no positive numbers then print out "Array contains no positive numbers", avoid any run-time errors.

```
double sum = 0.0;
int count = 0;

for (int i = 0; i < scores.length; i++)
   if (scores[i] > 0) {
      sum += scores[i];
      count++;
  }

if (count > 0)
    System.out.println("Average is " + sum/count);
else
    System.out.println("Array contains no positive numbers");
```

8. a) Assume you have an array of length 100 which has already been filled with data read in from the user. Assume this array is called `values`, and is an array of ints. Write code which finds the duplicates of the values in the `values` array, and reports both the duplicate value and the two indices of the array in which it was found. (ie. Found value 2 at indices 3 and 4)

```
for (int i = 0; i < values.length; i++)
   for (int j = i+1; j < values.length; j++)
       if (values[i] == values[j])
          System.out.println("Found value " + values[i] + " at indices " + i + " and " + j);
```

b) Write a while loop that repeatedly prompts the user for a number of type int, reads an integer number in from the keyboard. The loop should continue until the user enters value 0, and it should count how many **positive** and **negative** numbers the user entered. It should also keep track of **positive_sum** and **negative_sum**. After the loop, print out **positive_average** and **negative_average**.

```
int positive = 0, positive_sum = 0;
int negative = 0, negative_sum = 0;
int num;

System.out.print("Please enter an int ");
while ((num = input.nextInt()) != 0) {
    if (num > 0) {
        positive++;
        positive_sum += num;
   }

   if (num < 0) {
      negative++;
      negative_sum += num;
   }
   System.out.print("Please enter an int ");

}

if (positive > 0)
    System.out.println("Positive average" + positive_sum/positive);

if (negative > 0)
    System.out.println("Negative average" + negative_sum/negative);
```

9. Given the program shown below indicate the output in the box below. This is one continuous program. Follow each step carefully, you **may** want to draw the array created with values in each entry to help you answer this question. (**partial credit**)

```
public class ProblemNine {

        public static void main(String[] args) {

                int LITTLE = 5, MEDIUM = 10, BIG = 20;

                int i,j, n = 9,temp;
                int[] num= new int[BIG];

                num[0]=9; num[1]=3; num[2]=4; num[3]=8; num[4]=2; num[5]=1;
                num[6]=5; num[7]=6; num[8]=7; num[9]=10;


                num[LITTLE] = 1;
                i = MEDIUM;
                while (i < BIG) {
                        num[i] = 2*num[i % MEDIUM];
                        i++;
                }

                for (j=0;j<BIG;j++) {
                        if (j % LITTLE == 0)
                                System.out.println();
                        System.out.print(num[j]+", ");
                }
        }
}
```

```
9, 3, 4, 8, 2,
1, 5, 6, 7, 10,
18, 6, 8, 16, 4,
2, 10, 12, 14, 20,
```

Output

9	3	4	8	2	1	5	6	7	10
18	6	8	16	4	2	10	12	14	20

10. Find and fix the bug(s) in the following code. For each line, indicate the line number, whether it is (C)ompile-time, (R)un-time, or (L)ogical, it can have multiple errors so circle all relevant ones. In the last entry, give the corrected code or SAME if there are no errors and changes needed.

```
/*
* Calculate the sum of all the elements Squared from the array
*/

0 public static void main (String [] args) {
1    double[] inp = {1.0, 2.0. 3,.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0}
2    int sum;

3    for (int i = 1; i <= inp.length-1; ++i ) {
4       Sum = inp[i] + inp[i*i];
5    }
6    System.out.println(“Sum of Squares is ” + sum);

7 }
```

Line #	Error Type	Corrected Code
0	C L R	SAME
1	**C** L R	double[] inp = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0};
2	**C** **L** R	double sum = 0.0;
3	C L R	SAME
4	**C** **L** **R**	sum += inp[i] * inp[i];
5	C L R	SAME
6	C L R	SAME
7	C L R	

11. Count the number of iterations for each of the loops below, assume max = 20.

a. `for (int i = 0; i < max; i++)` _____20_____

b. `for (int i = 0; i <= max; i++)` _____21_____

c. `for (int i = max-1; i > 0; i--)` _____19_____

d. `for (int i = max-1; i < 0; i--)` _____0_____

e.
```
for (int i = 1; i <= max; i++)
   if (i == max/2)
      break;
```
_____10_____

f.
```
for (int i = 1; i <= max; i++)
   if (i == max/2)
      continue;
```
_____20_____

g. `for (int i = 0; i <= max; i += 4)` _____6_____

h. `for (int i = 0; i <= max; i *= 4)` _____infinite_____

i. `for (int i = 1; i <= max; i += 4)` _____5_____

j. `for (int i = 1; i <= max; i *= 4)` _____3_____

12. Answer if each of the following statements are valid or invalid. Invalid means Eclipse will flag it as an error.

```
int i = 0;                                          //    Valid       Invalid
int max = 10;
int count;
int[] arr = new int[max-1];                         //    Valid       Invalid

for (int i = 0; i < max; i++);                      //    Valid       Invalid
System.out.println(arr[i] = i);                     //    Valid       Invalid


while (i < max) {
      i += 2;                                       //    Valid       Invalid
      if (i % 2 == 0) {
            int j = 1;
            for (j = max-2; j >= 0; j--)            //    Valid       Invalid
                  System.out.println(arr[j]);
            count++;                                //    Valid       Invalid
      } else {

            int count = j;                          //    Valid       Invalid
            for (int j = max-2; j >= 0; j--)        //    Valid       Invalid
                  System.out.println(arr[j]);
            count--;                                //    Valid       Invalid
      }
}
```

count++ and count-- could be marked invalid due to no initialization when count was declared. So either answer would be okay for this problem as the statement themselves are fine.