Problem 1.a (declaration, combined assignment):
x = 6
y = 3

Problem 1.b (casting):
a = 3
b = 4
d = 3.5
e = 3.0

Problem 1.c (Boolean variable):
b=false
c=false
d = true

Problem 1.d (mainly on String +! indexof, substring, equals, length, not array length, charAt are at knowledge level)
s = “hellohello3”
t = “hello3”
c = 3

Problem 1.e (if decision structure),
left side:
2.5
4.5
right side:
7

Problem 1.f (Boolean value print-out):
the value 3>4 is: false

Problem 1.g (loop, for, while, do-while):
2
1
3
2
3
3

Problem 1.h (String):
“hello goodbye goodbye world”
“hello goodbye goodbyegoodbyegoodbye world”
“hello goodbye goodbyegoodbyegoodbyegoodbyegoodbye world”

Problem 1.i:
N = 4259
sum = 0
sumStr = ""

N = 425
sum = 9
sumStr = "9"

N = 42
sum = 14
sumStr = "95"

N = 4
sum = 16
sumStr = "952"

sum at the end = 20
sumStr at the end = "9524"

summary/description =
This loop adds up each digit in N, storing their sum in the variable sum. It also reverses the digits, in sumStr.

Problem 1.j:
a = [9, 11, 26, 19, 33, 49, 53, 59]
summary/description =
This code adds up the elements in the array.

Problem 1.k:
a = [2, 9, -7, 15, 16, 14, 6, 4]
summary/description =
This code swaps the order of pairs of elements in the array.

Problem 2.a

```
System.out.println("Please enter a positive integer:");
intnum = kb.nextInt();
System.out.println(
               "The possible factors of " + num + " are:");

        // I'm only writing it this way so it fits neatly
for(
               intpossibleFactor = 1;
               possibleFactor<= num;
               possibleFactor++
          )
        {
if(num % possibleFactor == 0) {
System.out.println(possibleFactor);
            }
        }
```

Problem 2.b (This was a bit longer than I intended, and longer than anything you'll see on the exam)

```
        // first, read in 20 doubles from the keyboard,
        // and store them in an array:
System.out.println("please enter 20 doubles:  ");
double [] someDoubles = new double[20];
for(inti=0; i<someDoubles.length; i++) {
someDoubles[i] = kb.nextDouble();
        }

        // second, find the position of the smallest number
intsmallestPosition = 0;    // position of smallest num
     // value of smallest num
     doublesmallestValue = someDoubles[0];

     // Accumulation loop:
for(inti=1; i<someDoubles.length; i++) {
if(someDoubles[i] <smallestValue) {
smallestPosition = i;
smallestValue = someDoubles[i];
            }
        }

        // swap the smallest number with the number at pos 0
double temp = someDoubles[0];
someDoubles[0] = smallestValue;
someDoubles[smallestPosition] = temp;
```

Problem 2.c

```
Scanner input = new Scanner(System.in);
intcurrentStep = 0;
int accumulator = 0; //(or double/String/it depends) (or 0.0,"")
while(currentStep< 10) {
                    // Counter loop with accumulation of sum
     intcurrent_value = input.nextDouble();
                    // something, it depends
     accumulator += current_value;
                    // ACCUMULATE(accumulator, current_value);
     currentStep ++;
}
double average = accumulator / 10.0;

System.out.println("average = " + average);
```

Problem 2.d

```
Scanner input = new Scanner(System.in);
intcurrentStep = 0;
int accumulator = 0;
while(currentStep< 10) {
                    // counter loop of accumulation of %2 check
     intcurrent_value = input.nextInt();
     //test to see if current_value is odd
     if(current_value % 2 ==  1) {
          accumulator = accumulator + 1;
     }
     currentStep ++;
}
System.out.println("number of odd values:  " + accumulator);
```

Problem 3

```
importjava.util.Scanner;

public class Steps
{
public static void main(String [] args)
    {
        Scanner kb = new Scanner(System.in);

System.out.println("Enter number of steps");
intnumSteps = kb.nextInt();

System.out.println("Enter width of steps");
int width = kb.nextInt();

System.out.println("Enter height of steps");
int height = kb.nextInt();
```

```
        // loop that does 1 iteration for each step
for(int step=0; step<numSteps; step++ )
        {
            // loop that prints the right number of spaces
            // before the start of the top of the step
for(int space=0; space<step*(width+1); space++)
            {
System.out.print(" ");
            }
            // loop that prints the top of the step
for(int hyphen = 0; hyphen < width; hyphen++)
            {
System.out.print("-");
            }
System.out.println();  // end of top of step

            // loop that does 1 iteration for each row
            // below the top of the step
for(int row=0; row<height; row++)
            {
                // inner loop that prints
                  // the right number of spaces for this row
for(int space=0;
                      space<(step+1)*(width+1)-1;
                      space++)
                {
System.out.print(" ");
                }
System.out.println("|"); // end of 1 row of step
            }
        }
    }
}
```