I. Fill in the Blank (30 pts)

1. If a variable in a class definition is declared **private**, then it can only be accessed in the class definition.

2. The **split** method in the String class will create an array containing tokens in the String separated by a delimiter.

3. We use the **length()** method to determine the number of characters in a String.

4. If an entity is declared **final**, then once it is assigned a value, its value can’t be changed.

5. If the characters // are encountered on a line, then the compiler ignores the rest of the line.

6. **Method overloading** occurs when we have two or more methods in a class definition with the same name but different parameter lists.

7. The **constructor** in a class definition always has the same name as the class.

8. **Pseudocode** are English like instructions where we spell out what we will do but it isn’t written in a form that a computer can understand.

9. An object created from a class is called **immutable** if there are no public means to change the variables defined in the class.

10. The numeric literals 1, 14, 22, 38, and 74 are automatically of type **int**.

11. An object becomes **garbage** when there are no valid references to it.

12. If a variable in a class definition is not declared static, then it is an **instance** variable.

13. All parameter passing in Java is by **value**.

14. **Information Hiding** is the ability to restrict the access to or the visibility of data.
15. Older languages allowed spaghetti code which is a perjorative term for a program with many goto statements.

16. A while loop is a pretest loop since the condition is checked before the body is executed.

17. **Encapsulation** is the binding together of data and the code that operates on the data.

18. If a continue statement is encountered in a loop, then the remainder of the current iteration is skipped.

19. We use the length field of an array to determine its size.

20. If the return type on a method is void, then the method doesn’t return a value.

21. The signature of a method in Java is a combination of its name and parameter list.

22. In short-circuit evaluation, if the value of a logical boolean expression can be determined by examining the first operand, then the second isn’t evaluated.

24. A paradigm is a style of programming.

25. The indexOf method in the String class determines the position of the first occurrence of a character in the String.

26. The scope of a variable is the portion of the program where the variable is known.

27. Java uses UNICODE to encode characters.

28. In order to change a double variable to an int, we perform a cast.

29. If a type is signed, then the value of the 1st bit in its binary value is significant.

30. If an array is named numbers, then the last valid position in the array is numbers.length – 1.
II. Matching (10 pts)

31. null H  A. Will cause all surrounding loops or switch statements to end

32. formal parameters K B. When used between object references, tests whether or not the contents of
    the objects they refer to are the same

33. final int[] numbers I C. When used between object references, tests whether or not they refer to the
    same objects

34. default constructor M D. String concatenation

35. break U E. Numerical addition

36. a++ L F. Has the value 3 if a is currently 2

37. Legal Identifiers in Java S G. The default value of a primitive type

38. Hardware R H. The default value of a reference variable

39. == C I. Creates a reference to an int[] and numbers cannot be reassigned to refer to
    another array

40. + in 1 + "" D J. Is generated anytime we use the new operator on a constructor of a class with
    no parameter list regardless of whether there are any constructors defined in the class

K. The parameters listed in a method declaration

L. Has the value 2 if a is currently 2

M. Is generated by the compiler when the new operator is invoked on a
    constructor of a class with no parameter list and there are no constructors
    defined in the class

N. Creates a reference to an int[] whose contents can’t be changed

O. name45, name$, name_, and name-

P. The software that controls the components in a computer.

Q. The parameters listed in a method call

R. The physical components in a computer.

S. name123, name45, name$, and name_

T. name123, name45, name$, and name-

U. Will cause the closest surrounding loop or switch statement to end
III. Short Answer (20 pts)

41. (2 pts) What is the binary equivalent of the decimal number 75?

\[ 75 = 1 \times 64 + 0 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 \] so the binary equivalent of 75 is 1001011

42. (2 pts) Given that an int is stored in 4 bytes and is a signed type, what are the smallest and largest int values that can be stored?

The smallest value is \(-2^{31}\), and the largest value is \(2^{31} - 1\)

43. (2 pts) Would the following cause a compile-time error? Explain.

```java
final int i = 1;
i = 2;
```

Yes, it would cause a compile-time error. Since i is declared final and we have assigned it a value, we can’t change its value

44. (2 pts) How would we generate a random integer between 192 and 291?

\[ 192 + \text{(int)}(99 \times \text{Math.random()}) \]

45. (2 pts) What would happen if the following three lines are executed?

```java
int i = 1;
while (i = 1)
    System.out.println(i);
```

There will be a compile-time error since the type of the condition in a while statement must be boolean
46. (2 pts) Does the Math.abs method always return a positive value? Explain.

No, in the case of the smallest int value, Math.abs returns a negative value

47. (2 pts) What is the syntax for creating an array in one line that consists of the even numbers between 2 and 12 with 2 and 12 included?

    int[] numbers = {2,4,6,8,10,12};

48. (2 pts) In what constructs is the break statement allowed?

    for loop, do while, while, and switch

49. (2 pts) What is the syntax for creating a void method called sum that can accept any number of char parameters?

    public void sum(char... characters)

50. (2 pts) Suppose we have an array called numbers containing ints. Show how the for each loop can be used to display the contents of the array.

    for (int number: numbers)
        System.out.println(number);
IV. Discussion (10 pts)

51. (3 pts) What is the implication of short-circuit evaluation? Give an example to illustrate this.

The implication of short-circuit evaluation is that the second operand in the expression is not evaluated so you cannot make any assumption about it.

In

if (i != 0 && ((j=i) < 1))

If i is equal to 0, then the value of j will not be changed in the second operand.

52. (3 pts) Explain how the Bubble Sort works.

We make several passes through the array each time comparing adjacent elements. The smaller elements will "bubble" to the top.

53. (4 pts) Explain what it means for an instance variable to “belong” to an instance and not to the class. Give an example to illustrate this.

It means that an instance variable cannot be accessed from a class method like the main method because instance variables only have meaning to instances.

```java
public class Test {
    private int num;

    public static void main(String[] args) {
        System.out.println(num);
    }
}
```

This would cause a compile-time error.
V. Problem Solving and Coding (30 pts)

54. (5 pts) What does the following print? Explain the output. You may use a calculator, abacus, or ENIAC computer if you have one available if you want.

```java
public class Question54 {
    public static void main(String[] args) {
        String temp = “This is a test”;
        temp.concat(“World” + temp.length() + temp.charAt(4) + temp.charAt(5));
        temp.concat(“T” + temp.charAt(temp.length()/2));
        temp.concat(“WOW this is alot” + temp.charAt(5));
        temp = “”;
        temp.concat(“J” + temp.charAt(temp.length()-4/5-3) + temp.indexOf(’s’));
        temp.concat(“K” + temp.charAt(temp.length()/(int)Math.sin(175*3.14/180)+5));
        temp.concat(“R” + temp.length()*(int)Math.pow(4,15)*(int)Math.sin(36*temp.length()));
        temp.toUpperCase().toLowerCase();
        System.out.println(temp.length());
        System.out.println(temp);
    }
}
```

0 followed by a blank line will be printed. We can’t change a String by calling a method on it so
the only line that affects temp is where temp is set to refer to an empty String.

55. (5 pts) What does the following code print? Explain the output.

```java
for (int counter=0;counter<40;counter++) {
    if (counter % 2 == 0 && counter % 3 == 0 && counter > 0)
        continue;
    if (counter % 4 == 0 && counter % 2 != 0)
        break;
    System.out.println(counter);
}
```

All integers from 0 to 40 except for the positive multiples of 6 are printed.
56. (5 pts) Suppose we want to sort an array of ints using the Selection Sort. Fill in the missing code.

```java
public class Question5 {
    public static void swap(int[] array,
                            int firstIndex,
                            int secondIndex) {
        int temp = array[firstIndex];
        array[firstIndex] = array[secondIndex];
        array[secondIndex] = temp;
    }
    public static void sort(int[] array) {
        for (int counter=0; counter < array.length-1; counter++)
            for (int counter1=1; counter1 < array.length; counter1++)
                if (array[counter] > array[counter1])
                    swap(array, counter, counter1);
    }
    public static void main(String[] args) {
        int[] numbers = {22,14,18,5,6,12,10};
        sort(numbers);
    }
}
```
public class Question57 {
    public static void count(String input) {
        int[] count = new int[256];
        for (int counter=0;counter<input.length();counter++)
            count[(int)input.charAt(counter)]++;
        for (int counter=0;counter<count.length;counter++)
            if (input.indexOf((char)counter) != -1)
                System.out.println((char)counter + " " + count[counter]);
    }
    public static void main(String[] args) {
        count("Test");
    }
}

public class Question58 {
    public static int[] addToArray(int[] numbers, int number) {
        int[] temp = new int[numbers.length+1];
        System.arraycopy(numbers,0,temp,0,numbers.length);
        temp[numbers.length] = number;
        return(temp);
    }
    public static void main(String[] args) {
        int[] numbers = {1,2,3,4,5};
        numbers = addToArray(numbers,1);
        System.out.println(java.util.Arrays.toString(numbers));
    }
}