I. Fill in the Blank (30 pts)

1. In order to convert a double to an int, we must use **casting**.

2. The parameters in the call to a method are called the **actual** parameters.

3. An int is a **signed** data type because the first bit indicates something special about what is stored in the int.

4. **Method overloading** involves having more than one method in a class definition with the same name.

5. The **scope** of a variable is the portion of code in which it is accessible.

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

7. A Java identifier can contain letters, digits, $, and _, but cannot start with a **digit**.

8. **UNICODE** is used to encode characters in Java.

9. If the two operands in an operation are ints, the type of the result is **int**.

10. If we create an array of ints, the initial value of each int is **0**.

11. Java uses pass by **value** to send parameters to a method.

12. A **nested** loop is one that is inside another loop.

13. The compiler will ignore any part of a line of code after //.

14. **32** bits are used to store an int in Java.

15. We use the operator **==** to test for equality between two primitive types.
16. If a **continue** statement is encountered in a loop, then the remainder of that iteration is skipped.

17. **Hardware** are the components of a computer that we can physically touch.

18. **java.lang** is automatically imported into every program in Java.

19. The three types of errors we encounter in programming are **compile-time**, run-time, and logic.

20. An array is somewhat an example of an **immutable** object because its size can’t be changed after it is created although its contents can be changed.

21. If a variable is declared **final**, we cannot assign a value to it more than once.

22. If a method does not return a value, we should make its return type **void**.

23. A **package** in Java is a collection of related classes.

24. The type of the condition in an if or while statement must be **boolean**.

25. Structured programming tries to avoid the use of the **goto** statement.

26. We obtain the size of an array by accessing the field **length** associated with the array.

27. The **Binary** search technique involves examining the midpoint of a sorted array to locate an element.

28. True/False **False** If the return type of a method is double[], we can return an int[].

29. The index of the first element in an array is **0**.

30. The parameters in the declaration of a method are called **formal** parameters.
II. Matching (10 pts)

31. != P
   A. An environment variable that contains a list of directories that will be searched for Java class files

32. pre-test loop N
   B. array.size-1

33. import R
   C. Illustrated by DayOfWeek

34. The last valid position G
   D. Makes several passes through an array each time holding an element in place and comparing it with each remaining element in an array

35. Scanner L
   E. A loop in which a condition is checked after the body of the loop is executed

36. Reassigns an already declared S
   F. A statement that is used to declare that our Java program is part of another class
   variable to point to a new array

37. CamelCase C
   G. array.length-1

38. CLASSPATH A
   H. array.length

39. Bubble Sort I
   I. Makes several passes through an array each time comparing each pair of adjacent elements

40. A valid declaration of a method Q
   J. A device Java uses to interpret handwriting of variable arity
   of variable arity

K. array = {1,2,3,4,5};

L. The class we use to read input from the keyboard

M. The operator used to emphasize assignment

N. A loop in which a condition is checked before the body of the loop is executed

O. public int sum(int... array,int num)

P. The operator used to test for inequality

Q. public int sum(int... array)

R. The statement we use to include another Java class in our program

S. array = new int[] {1,2,3,4,5};

T. Illustrated by dayofweek

U. An environment variable that contains a list of directories that will be searched for executable programs
III. Short Answer (20 pts)

41. (3 pts) What is the binary equivalent of the decimal number 51?

\[ 51 = 1 \times 32 + 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \] so the equivalent is 110011

42. (3 pts) If we use four bits to store a number, what is the representation of -5?

5 = 0101 we flip the bits and add 1 so -5 = 1011

43. (3 pts) Is there a difference in what the following loops print? If so, why is there a difference?

Loop 1
\begin{verbatim}
int counter = 0;
while (counter++ < 10)
    System.out.println(counter);
\end{verbatim}

Loop 2
\begin{verbatim}
int counter = 0;
while (++counter < 10)
    System.out.println(counter);
\end{verbatim}

Yes, there is a difference. As an expression counter++ is the value of counter before the increment and ++counter is the value of counter after the increment. So the second loop will print out one less number than the first.

44. (3 pts) Recall that expressions are evaluated left to right, what is the value of the following expression?

\[ \frac{4}{5} \times 15 \times 530 \times \frac{837}{9240} \times 1500 - 3 \times 5 + \frac{2}{15} \times 4 \]

The first part of the expression is composed of multiplication and division which are of equal precedence. Since the first division \( \frac{4}{5} \) produces 0, the remainder of the products and quotients are irrelevant. The second expression evaluates to -15. The last expression evaluates to 0. So the value of the expression is \( 0 - 15 + 0 = -15 \).
45. (3 pts) What is the output of the following? Explain the output.

```java
System.out.println(1 + 2 + "Hello" + 3 + 4);
```

Again, expressions are evaluated left to right. The first two operands are ints, so the result is an int. 3 is the value of the first expression. Since the next operand is a String, the + operator becomes String concatenation. From that point on, + is String concatenation. So the result is 3Hello34

46. (3 pts) What is the value of found after the following statement executes?

```java
boolean found = (15 < 30/2) ? (5*4 % 3 == 0) : (6*5-2 > 5);
```

The output of the ternary operator is based on the condition. Since the condition is false, the value of the expression will be the value following the :. The expression 6*5-2 > 5 is true. So the value of found is true.

47. (2 pts) What line of code will produce a random integer between 100 and 200 with 200 not included?

```java
100 + (int)(100*Math.random())
```
IV. Discussion (10 pts)

48. (5 pts) What are the steps in the Polya Problem Solving Methodology?

1. Understand the problem
2. Design or devise a good plan to solve the problem. We call this developing an algorithm.
3. Implement the plan
4. Evaluate the implementation for its correctness and quality

49. (5 pts) Suppose we have a method which accepts an array as parameter. What can we do in the method that would prevent us from accessing the original array referred to by the actual parameter in the method call?

If we change what array the formal parameter refers to, then we can’t change the array referred to by the actual parameter.
V. Problem Solving and Coding (30 pts)

51. (5 pts) Will the following produce a compile-time error? Why or why not?

```java
public static boolean greaterThan400(int number) {
    if (number > 400 || number > 500)
        return(true);
    else if (number < 400 || number < 340)
        return(true);
}
```

Yes, there will be a compile-time error. Because there are multiple paths of execution through the method, we must make sure there is a valid return in each one. Because we don’t specify what should happen in the case where number is equal to 400, there is a compile-time error.

52. (5 pts) Fill in the missing pieces of this code so that the method will return a reference to a new int array containing the contents of the original array together with the number sent in as parameter.

```java
public static int[] add(int[] array,int number) {
    int[] temp = new int[array.length + 1];
    System.arraycopy(array,0,temp,0,array.length);
    temp[array.length] = number;
    return(temp);
}
```

53. (5 pts) What is produced by the following code? Explain the output.

```java
public class Question53 {
    public static void place(int[] array,int index,int number) {
        array = new int[array.length];
        array[index] = number;
    }
    public static void main(String[] args) {
        int[] array = {1,2,3,4,5};
        place(array,0,4);
        place(array,1,5);
        place(array,2,3);
        place(array,3,1);
        place(array,4,2);
        for (int number : array)
            System.out.println(number);
    }
}
```

The numbers 1 2 3 4 5 are printed out on separate lines. Because we change what array the formal parameter refers to in the method, it has no effect on the original array.
54. (7.5 pts) Suppose we want to write a method that will accept any number of int parameters and return the product of the parameters. Show the declaration and body of the method.

```java
public static int product(int ... array) {
    int product = array[0];
    for (int number : array)
        product *= number;
    return(product);
}
```

55. (7.5 pts) Suppose we have a method that accepts an int[] as parameter. Show the contents of the method so that the contents of the array will be reversed. That is, if 1,2,3,4,5 is the array sent to the method, after the method executes, the contents will be 5,4,3,2,1.

```java
public static void reverse(int[] array) {
    for (int counter=0;counter<array.length/2;counter++) {
        int temp = array[counter];
        array[counter] = array[array.length-1-counter];
        array[array.length-1-counter] = temp;
    }
}
```