1 Short Answer (15 Points Each)

1. Write a method called `FlipCount` that takes in one integer parameter `trials`. The method should simulate the flipping of a coin, `trials` times, and count the number of flips that result in Heads. The method should return that count.

   Solution:

```java
public static int FlipCount(int trials) {
    int count = 0;

    for (int i = 0; i < trials; i++) {
        int coin = (int) (Math.random() * 2);
        if (coin == 1)
            count++;
    }

    return count;
}
```

2. Write a method called `ExtractLast` that takes a single string as a parameter, extracts the last word from the string and returns that word.

   Solution:

```java
public static String ExtractLast(String s) {
    int pos = s.lastIndexOf(" ");
    String sub = s.substring(pos + 1);
    return sub;
}
```

3. Write a method called `Divisors` that takes in a single integer parameter `n`. The method should print out all of the divisors of the number n. Recall that a number k is a divisor of n if $1 \leq k \leq n$ and $\frac{n}{k}$ is an integer, that is, k divides evenly into n.

   Solution:

```java
public static void Divisors(int n) {
    for (int i = 1; i <= n; i++) {
        if (n % i == 0)
            System.out.print(i + " ");
    }
}
```

4. If we take n objects and find all the ways to select k objects from that set, we call that a combination and denote it as $\left(\begin{array}{c} n \\ k \end{array}\right)$, the formula for this calculation is $\left(\begin{array}{c} n \\ k \end{array}\right) = \frac{n!}{k!(n-k)!}$.

   Write a two methods, one called `Factorial` that takes a single integer parameter `n` and returns the factorial of `n`. The factorial of a number n is defined to be $n! = 1 \cdot 2 \cdot 3 \cdots n$, where we define $0! = 1$ and if the value of n is less than 0 simply have the method return -1. Then create another method called `Combination` that takes two integer parameters n and k and returns the value of the combination. Have the `Combination` method call the `Factorial` method to compute the factorials.

   Solution:

```java
public static int Factorial(int n) {
    if (n < 0)
        return -1;

    int fact = 1;
    for (int i = 1; i <= n; i++)
        fact *= i;
    return fact;
}

public static int Combination(int n, int k) {
    return Factorial(n) / (Factorial(k) * Factorial(n - k));
}
```

2 Program Traces (15 Points Each)

1. For each of the given inputs, write the output of the program.

```java
import java.util.Scanner;

public class Exam2Trace1 {

    public static int mth1(int b, int c, int a) {
        System.out.println("In Method 1");
        System.out.println(a + " " + b + " " + c);
        return a - b * c;
    }

    public static int mth2(int a, int b, int c) {
        System.out.println("In Method 2");
        System.out.println(a + " " + b + " " + c);
        if (a > b)
            return mth3(a, b, c);
        else
            return mth3(b, a, c);
    }

    public static int mth3(int c, int b, int a) {
        System.out.println("In Method 3");
        System.out.println(a + " " + b + " " + c);
        return mth1(a, b, c);
    }

    public static int mth4(int b, int a, int c) {
        System.out.println("In Method 4");
        System.out.println(a + " " + b + " " + c);
        if (a > b && b > c)
            return a;
        else if (a > b)
            return c;
        else
            return mth2(c, b, a);
    }

    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("Input: ");
        int a = kb.nextInt();
        int b = kb.nextInt();
        int c = kb.nextInt();

        System.out.println(mth1(a, b, c));
        System.out.println();
        System.out.println(mth2(a, b, c));
        System.out.println();
        System.out.println(mth4(a, b, c));
    }
}
```

(a) `Input: 7 5 3`

Solution:

```
In Method 1
3 7 5
-32

In Method 2
7 5 3
In Method 3
3 5 7
In Method 1
7 3 5
-8

In Method 4
5 7 3
In Method 2
3 7 5
In Method 3
5 3 7
In Method 1
7 5 3
-8
```

(b) `Input: 5 7 3`

Solution:

```
In Method 1
3 5 7
-32

In Method 2
5 7 3
In Method 3
3 5 7
In Method 1
7 3 5
-8

In Method 4
7 5 3
7
```

2. For each of the given inputs, write the output of the program.

```java
import java.util.Scanner;

public class Exam2Trace2 {

    public static String DoSomething(String str1, String str2, int p) {
        str1 += " ";
        int c = 0;
        int pos = -1;
        while (c < p) {
            pos = str1.indexOf(str2, pos + 1);
            if (pos >= 0)
                c++;
            else
                return "Error";
        }
        c = pos;
        while (str1.charAt(c) != ' ') {
            c--;
        }
        c++;
        pos = str1.indexOf(" ", c);
        return str1.substring(c, pos);
    }

    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("Input String: ");
        String s1 = kb.nextLine();
        System.out.print("Input String: ");
        String s2 = kb.nextLine();
        System.out.print("Input Number: ");
        int a = kb.nextInt();
        System.out.print(DoSomething(s1, s2, a));
    }
}
```

(a)
```
Input String: A program is simply a list of unambiguous instructions
Input String: i
Input Number: 5
```
Solution:

```
instructions
```

Values of pos in the loop: 10, 14, 23, 35, 42

Final value of c: 42

(b)
```
Input String: A program is simply a list of unambiguous instructions
Input String: s
Input Number: 3
```
Solution:

```
list
```

Values of pos in the loop: 11, 13, 24

Final value of c: 22

3 Coding (20 Points)

Do one and only one of the following exercises.

1. Write a program that will simulate tossing a coin repeatedly until you get a run of heads of a given size. That is, a run of 2 would be tossing HH, a run of 3 would be tossing HHH consecutively, a run of 4 would be tossing HHHH consecutively, and so on. Have the program count the number of rolls needed for each possible run from 1 to 20. The output should look like the following.

```
Number of coin tosses for a run of 1 heads = 1
Number of coin tosses for a run of 2 heads = 12
Number of coin tosses for a run of 3 heads = 16
Number of coin tosses for a run of 4 heads = 92
Number of coin tosses for a run of 5 heads = 43
Number of coin tosses for a run of 6 heads = 223
Number of coin tosses for a run of 7 heads = 469
Number of coin tosses for a run of 8 heads = 742
Number of coin tosses for a run of 9 heads = 698
Number of coin tosses for a run of 10 heads = 3353
Number of coin tosses for a run of 11 heads = 726
Number of coin tosses for a run of 12 heads = 5507
Number of coin tosses for a run of 13 heads = 30778
Number of coin tosses for a run of 14 heads = 34318
Number of coin tosses for a run of 15 heads = 12447
Number of coin tosses for a run of 16 heads = 48803
Number of coin tosses for a run of 17 heads = 578673
Number of coin tosses for a run of 18 heads = 134327
Number of coin tosses for a run of 19 heads = 185188
Number of coin tosses for a run of 20 heads = 414894
```

Solution:

```java
public class Exam2Program1 {

    public static void main(String[] args) {
        for (int run = 1; run <= 20; run++) {
            boolean done = false;
            long count = 0;
            int runcount = 0;
            while (!done) {
                int roll = (int) (Math.random() * 2);
                if (roll == 1)
                    runcount++;
                else
                    runcount = 0;

                count++;
                if (runcount == run)
                    done = true;
            }

            System.out.println("Number of coin tosses for a run of " + run + " heads = " + count);
        }
    }
}
```

2. Write a program that will take an input string from the user and convert the string to Pig Latin. Pig Latin is a language game in which words in English are altered by the following rules.

   (a) For words that begin with consonants, all letters before the initial vowel are placed at the end of the word sequence. Then, "ay" is added, as in the following examples:

   - pig → igpay
   - banana → ananabay
   - trash → ashtray
   - happy → appyhay
   - duck → uckday
   - glove → oveglay
   - thanks → anksthay
   - will → illway

   (b) For words that begin with vowels, one just adds "way" to the end. Examples are:

- eat → eatway
- omelet → omeletway
- are → areway
- egg → eggway

The main program should take the input string and extract each word of the string, one by one. It should then call the a method, `PigLatinWord`, that takes in a string, assumed to be a single word, and converts the word to Pig Latin. You may assume that there is no punctuation in the string and consider 'y' to be a vowel. The main program should also print out the original phrase and the Pig Latin conversion. The output should look like the following.

```
Input String: Methods are also known as functions and subroutines
Pig Latin: ethodsMay areway alsoway ownknay asway unctionsfay andway ubroutinessay
```

Solution:

```java
import java.util.Scanner;

public class Exam2Program1 {

    public static String PigLatinWord(String str) {
        if (str.startsWith("a") || str.startsWith("e") || str.startsWith("i") || str.startsWith("o")
                || str.startsWith("u") || str.startsWith("y"))
            return str + "way";

        int firstvowel = str.length();
        int i = 0;
        while (i < firstvowel && i < str.length()) {
            if (i < firstvowel) {
                char ch = str.charAt(i);
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' || ch == 'y')
                    firstvowel = i;
            }
            i++;
        }

        String begin = str.substring(0, firstvowel);
        String end = str.substring(firstvowel);
        return end + begin + "ay";
    }

    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("Input String: ");
        String s = kb.nextLine();
        s = s + " ";
        String PLString = "";
        int pos = 0;
        while (pos >= 0) {
            int start = pos;
            pos = s.indexOf(" ", pos + 1);
            if (pos >= 0) {
                String word = s.substring(start, pos);
                word = word.trim();
                String PLword = PigLatinWord(word);
                PLString += PLword + " ";
            }
        }
        System.out.println("Pig Latin: " + PLString);
    }
}
```