Lab 1

1)
// a. Write a method isPrime to accept one integer parameter and to check whether that parameter is prime or not.

// b. Using this method, generate first N prime numbers in the main method.

```
import java.util.Scanner;

class Prime{
	boolean isPrime(int n){
		for(int i = 2; i < n; i++){
			if(n % i == 0)
				return false;
		}
		return true;
	}

	public static void main(String []args){
		Prime p = new Prime();
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		for(int i = 1; i <= n; i++){
			if(p.isPrime(i))
				System.out.println(i);
		}
	}
}
```

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab1$ javac prime.java
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab1$ java Prime
10
1
2
3
5
7
```

2)
// Arrange the elements in ascending and descending order using Bubble sort method.

```
import java.util.Scanner;
class Bubble{

	void display(int a[]){
		int n = a.length;
		for(int i = 0; i < n; i++){
			System.out.print(a[i] + " ");
		}
		System.out.println();
	}

	void bubblesort(int a[], boolean ascending){
		int n = a.length;
```

```
                for(int i = 0; i < n; i++){
                        for(int j = 0; j < n-1-i; j++){
                                if(ascending){
                                        if(a[j] > a[j+1]){
                                                int temp = a[j];
                                                a[j] = a[j+1];
                                                a[j+1] = temp;
                                        }
                                }
                                else{
                                        if(a[j] < a[j+1]){
                                                int temp = a[j];
                                                a[j] = a[j+1];
                                                a[j+1] = temp;
                                        }
                                }
                        }
                }
                display(a);
        }
        public static void main(String []args){
                Bubble b = new Bubble();
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter length and elements");
                int n = sc.nextInt();
                int a[] = new int[n];
                for(int i = 0; i < n; i++)
                        a[i] = sc.nextInt();
                System.out.println("Ascending: ");
                b.bubblesort(a, true);
                System.out.println("Descending: ");
                b.bubblesort(a, false);
        }
}
```

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab1$ javac bubble.java
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab1$ java Bubble
Enter length and elements
6
3
5
1
6
7
9
Ascending:
1 3 5 6 7 9
Descending:
9 7 6 5 3 1
```

3)

```
// Find the addition of two matrices and display the resultant matrix.

import java.util.Scanner;
import java.lang.Math;
```

```
class AddMat{
   public static void main(String arg[]){
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter matrix height: ");
      int rows=sc.nextInt();
      System.out.print("Enter matrix width: ");
      int cols=sc.nextInt();
      int matrix1[][]=new int[rows][cols];
      int matrix2[][]=new int[rows][cols];

      System.out.println("Matrix 1: ");
      for(int i=0; i<rows; i++){
         for(int j=0; j<cols; j++){
            matrix1[i][j]=sc.nextInt();
         }
      }

      System.out.println("Matrix 2: ");
      for(int i=0; i<rows; i++){
         for(int j=0; j<cols; j++){
            matrix2[i][j]=sc.nextInt();
         }
      }

      for(int i=0; i<rows; i++){
         for(int j=0; j<cols; j++){
            matrix2[i][j]=matrix1[i][j]+matrix2[i][j];
            System.out.print(matrix2[i][j]+" ");
         }
         System.out.println();
      }
   }
}
```

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab1$ javac addmat.java
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab1$ java AddMat
Enter matrix height: 2
Enter matrix width: 2
Matrix 1:
1
2
3
4
Matrix 2:
5
6
7
8
6 8
10 12
```