Q1:

Code:

```
import java.util.Scanner;
public class q1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[][] A = new int[4][4];
        int[][] B = new int[4][4];
        int[][] AB = new int[4][4];
        int[][] BA = new int[4][4];
        System.out.println("Enter elements of matrix A (4x4):");
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                A[i][j] = scanner.nextInt();
            }
        }
        System.out.println("Enter elements of matrix B (4x4):");
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                B[i][j] = scanner.nextInt();
            }
        }
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                AB[i][j] = 0;
                for (int k = 0; k < 4; k++) {
                    AB[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        for (int i = 0; i < 4; i++) {
```

```
            for (int j = 0; j < 4; j++) {
                BA[i][j] = 0;
                for (int k = 0; k < 4; k++) {
                    BA[i][j] += B[i][k] * A[k][j];
                }
            }
        }
        System.out.println("Matrix AB:");
        printMatrix(AB);
        System.out.println("Matrix BA:");
        printMatrix(BA);
        boolean isEqual = true;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if (AB[i][j] != BA[i][j]) {
                    isEqual = false;
                    break;
                }
            }
        }
        if (isEqual) {
            System.out.println("AB is equal to BA.");
        } else {
            System.out.println("AB is not equal to BA.");
        }
        scanner.close();
    }
    private static void printMatrix(int[][] matrix) {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                System.out.print(matrix[i][j] + " ");
```

```
        }
        System.out.println();
      }
    }
}
```

Output:

```
PS D:\Mca_upes\codes and stuff> cd "d:\Mca_upes\codes and stuff\class
CA_Codings\java\Assignment-1\" ; if ($?) { javac q1.java } ; if ($?)
java q1 }
Enter elements of matrix A (4x4):
1 2 3 4
2 3 4 4
2 3 4 5
22 3 3 3
Enter elements of matrix B (4x4):
3 1 1 1
1 0 1 1
3 3 3 3
4 4 4 4
Matrix AB:
30 26 28 28
37 30 33 33
41 34 37 37
90 43 46 46
Matrix BA:
29 15 20 24
25 8 10 12
81 33 42 48
108 44 56 64
AB is not equal to BA.
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java\Assignment-1>
```

Algo:

1. **Get User Input**:
    - Prompt the user to enter elements of matrix A
    - Read the user's input using a **Scanner** object and store it in matrix A
    - Prompt the user to enter elements of matrix B
    - Read the user's input using a **Scanner** object and store it in matrix B
2. **Multiply Matrices**:
    - Initialize matrices AB and BA to store the product of A and B, and B and A, respectively
    - Iterate through each element of matrices A and B, and calculate the dot product of corresponding rows and columns
    - Store the results in matrices AB and BA
3. **Print Matrices**:
    - Print the elements of matrices AB and BA using a helper function **printMatrix**
4. **Check Equality**:
    - Initialize a boolean variable **isEqual** to **true**
    - Iterate through each element of matrices AB and BA, and check if they are equal
    - If any element is not equal, set **isEqual** to **false** and break the loop

Q2:

Code:

```
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter an integer: ");
      int number = scanner.nextInt();
      boolean isPrime = true;
      if (number <= 1) {
         isPrime = false;
      } else {
         for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
               isPrime = false;
               break;
            }
         }
      }
      if (isPrime) {
         int n = number;
         int binary[] = new int[32];
         int index = 0;
         while (n > 0) {
            binary[index++] = n % 2;
            n /= 2;
         }
         System.out.print("Binary representation: ");
         for (int i = index - 1; i >= 0; i--) {
            System.out.print(binary[i]);
         }
```

```
            System.out.println();
        } else {
            int n = number;
            int octal[] = new int[32];
            int index = 0;
            while (n > 0) {
                octal[index++] = n % 8;
                n /= 8;
            }
            System.out.print("Octal representation: ");
            for (int i = index - 1; i >= 0; i--) {
                System.out.print(octal[i]);
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

Output:

```
PS D:\Mca_upes\codes and stuff> cd "d:\Mca_upes\codes
CA_Codings\java\Assignment-1\" ; if ($?) { javac q2.ja
java q2 }
Enter an integer: 17
Binary representation: 10001
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java\
```

```
PS D:\Mca_upes\codes and stuff> cd "d:\Mca_upes\codes and s
CA_Codings\java\Assignment-1\" ; if ($?) { javac q2.java }
java q2 }
Enter an integer: 22
Octal representation: 26
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java\Assig
```

Algo:

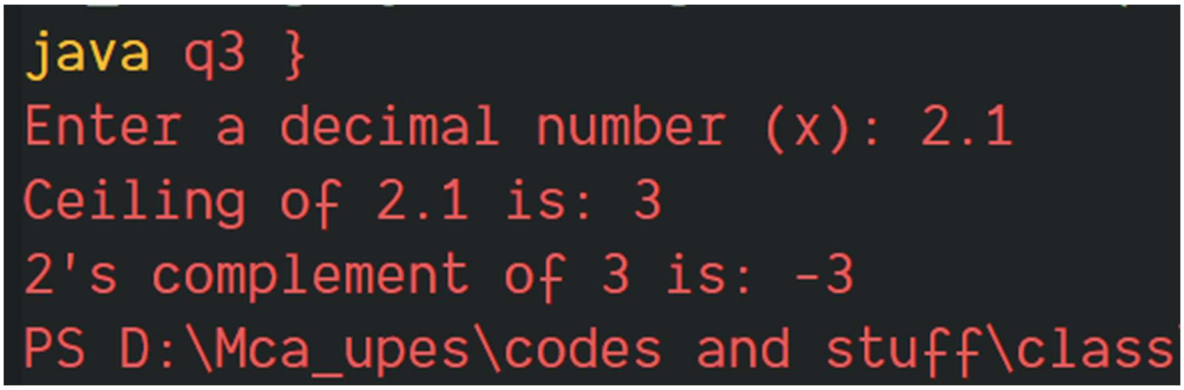
1. **Get User Input**:
    - Prompt the user to enter an integer
    - Read the user's input using a **Scanner** object
2. **Check if Prime**:
    - Initialize a boolean variable **isPrime** to **true**
    - If the input number is less than or equal to 1, set **isPrime** to **false**
    - Otherwise, iterate from 2 to the square root of the input number and check if it is divisible by any of these numbers
    - If it is divisible, set **isPrime** to **false** and break the loop
3. **Convert to Binary or Octal**:
    - If the input number is prime, convert it to binary representation
        - Initialize an array **binary** to store the binary digits
        - Iterate through the input number, dividing it by 2 and storing the remainder in the **binary** array
        - Print the binary representation by iterating through the **binary** array in reverse order
    - If the input number is not prime, convert it to octal representation
        - Initialize an array **octal** to store the octal digits
        - Iterate through the input number, dividing it by 8 and storing the remainder in the **octal** array
        - Print the octal representation by iterating through the **octal** array in reverse order

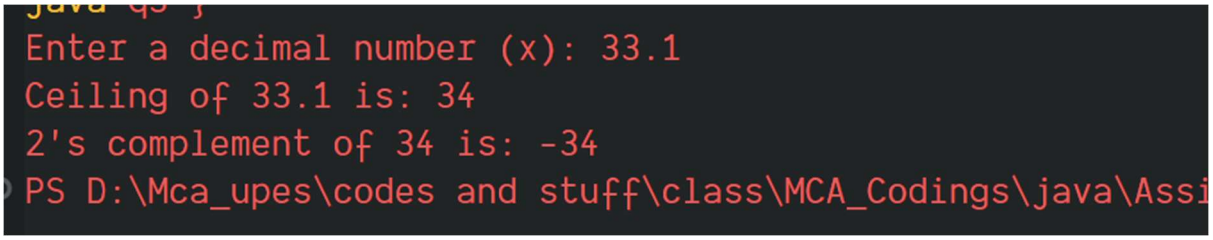
Q3:

Code:

```
import java.util.Scanner;
public class q3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a decimal number (x): ");
        double x = scanner.nextDouble();
        int y = (int) Math.ceil(x);
        System.out.println("Ceiling of " + x + " is: " + y);
        int twosComplement = ~y + 1;
        System.out.println("2's complement of " + y + " is: " + twosComplement)
        scanner.close();
    }
}
```

Output:





Algo:

Steps:

1. Get User Input:
    - Prompt the user to enter a decimal number (x)

- Read the user's input using a Scanner object

2. Calculate Ceiling:
    - Calculate the ceiling of x using the Math.ceil() method
    - Cast the result to an integer (y)
3. Print Ceiling:
    - Print the ceiling value (y) to the console
4. Calculate 2's Complement:
    - Calculate the 2's complement of y using the formula ~y + 1
5. Print 2's Complement:
    - Print the 2's complement value to the console

Q4

Code:

```
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;


class Flight {
   // private attributes
   private static int flightNumberCounter = 1000;
   private int flightNumber;
   private String destination;
   private LocalTime departureTime;
   private LocalTime ArrivalTime;
   // Constructor to assign values
   Flight(String destination, LocalTime arrival, LocalTime depart) {
      this.ArrivalTime = arrival;
      this.departureTime = depart;
      this.destination = destination;
      this.flightNumber = ++flightNumberCounter;
   }
   // private methods
   boolean upcomingFlight() {
      return departureTime.isAfter(LocalTime.now());
      // this function return true if departure time is more than current time
   }
   public boolean isComplete(){
      return ArrivalTime.isBefore(LocalTime.now());
      // this method return true if arrival time is less than current time
   }
   public int getFlightNumber(){
```

```
        return flightNumber;
    }
    // public method
    public void updateDeparture(LocalTime newDepart) {
        // updated departure time only get updated whn it is called
        this.departureTime = newDepart;
    }
    public void updateArrival(LocalTime newArrival) {
        // updated arrival time only get updated whn it is called
        this.ArrivalTime = newArrival;
    }
    public void display() {
        // to display details of flight,including flight number,destination and times
        System.out.println("Flight Number " + this.flightNumber);
        System.out.println("Flight Destination " + this.destination);
        System.out.println("Flight Departure time " + this.departureTime);
        System.out.println("Flight Arrival time " + this.ArrivalTime);
    }
}
class airport {
    // private attributes
    private String name;
    private List<Flight> flights;
    // Constructor to get value
    airport(String name){
        this.name=name;
        this.flights=new ArrayList<>();
    }
    // private methods
    private Flight findFlightByNumber(int flightNumber){
        for(Flight flight:flights){
```

```
        if(flight.getFlightNumber()==flightNumber){
            return flight;
        }
    }
    return null;
}
// public methods
public void addFlight(Flight flight) {
    flights.add(flight);
    // adds flight object to array list or add flight to airport
}
public void removeFlight(int flightNumber) {
    Flight flight=findFlightByNumber(flightNumber);
    if (flight!=null) {
        flights.remove(flight);
        System.out.println("Flight Number: "+flightNumber+" removed");
    }
    else{
        System.out.println("Flight Not Found");
    }
}
public List<Flight> upcomingFlights() {
    List<Flight>upcomingFlights=new ArrayList<>();
    for(Flight flight:flights){
        if(flight.upcomingFlight()){
            upcomingFlights.add(flight);
        }
    }
    return upcomingFlights;
}
public List<Flight> CompletedFlights() {
```

```
        List<Flight>completedFlights=new ArrayList<>();
        for(Flight flight:flights){
            if(flight.isComplete()){
                completedFlights.add(flight);
            }
        }
        return completedFlights;
    }
    public void displayDetails() {
        System.out.println("All flights at "+name+": ");
        for(Flight flight:flights){
            flight.display();
        }
    }

}

public class airportMain {
    public static void main(String[] args) {
        // Creating object for Airport
        airport airport=new airport("Demonic Ports");
        // Creating object for flights
        Flight flight1 = new Flight("New York", LocalTime.of(23, 30), LocalTime.of(8, 30));
        Flight flight2 = new Flight("London", LocalTime.of(15, 0), LocalTime.of(20, 0));
        Flight flight3 = new Flight("Paris", LocalTime.of(10, 0), LocalTime.of(12, 0));
        Flight flight4 = new Flight("India", LocalTime.of(3, 01), LocalTime.of(2, 0));
        // adding flights to airport
        airport.addFlight(flight1);
        airport.addFlight(flight2);
        airport.addFlight(flight3);
        airport.addFlight(flight4);
```

```
        // displaying all flight which are at airport
        airport.displayDetails();
        // updating arrival time of flight 1
        flight1.updateArrival(LocalTime.of(16, 10));
        System.out.println("\nAfter updating arrival time \n");
        airport.displayDetails();
        // removing flight from airport
        airport.removeFlight(flight2.getFlightNumber());
        // displaying remaining flight
        System.out.println("\nAfter removing flight 2\n");
        airport.displayDetails();
        // list of upcoming flights based on current time
        System.out.println("\nList of upcoming flights\n");
        for(Flight flight:airport.upcomingFlights()){
            flight.display();
        }
        // list of completed flights
        System.out.println("\n List of completed flights\n");
        for(Flight flight:airport.CompletedFlights()){
            flight.display();
        }

    }
}
```

Output:

```
PS D:\Mca_upes\codes and stuff> cd "d:\Mca_upes\codes and stuff\class\MCA_Codings\java\Assignment-1\" ; if ($?) { javac airportMain.java } ; if ($?) { java airportMain }
All flights at Demonic Ports:
Flight Number 1001
Flight Destination New York
Flight Departure time 14:30
Flight Arrival time 18:30
Flight Number 1002
Flight Destination London
Flight Departure time 15:00
Flight Arrival time 20:00
Flight Number 1003
Flight Destination Paris
Flight Departure time 10:00
Flight Arrival time 12:00

After updating arrival time

All flights at Demonic Ports:
Flight Number 1001
Flight Destination New York
Flight Departure time 14:30
Flight Arrival time 16:10
Flight Number 1002
Flight Destination London
Flight Departure time 15:00
Flight Arrival time 20:00
Flight Number 1003
Flight Destination Paris
Flight Departure time 10:00
Flight Arrival time 12:00
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java\Assignment-1>
```

```
After removing flight 2

All flights at Demonic Ports:
Flight Number 1001
Flight Destination New York
Flight Departure time 14:30
Flight Arrival time 16:10
Flight Number 1003
Flight Destination Paris
Flight Departure time 10:00
Flight Arrival time 12:00
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java\Ass
```

```
List of upcoming flights

Flight Number 1001
Flight Destination New York
Flight Departure time 08:30
Flight Arrival time 16:10
Flight Number 1003
Flight Destination Paris
Flight Departure time 12:00
Flight Arrival time 10:00
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java
```

```
 List of completed flights

Flight Number 1004
Flight Destination India
Flight Departure time 02:00
Flight Arrival time 03:01
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\
```

Algo:

Step 1: Define Classes

1. **Create `Flight` Class**
   - **Attributes**:
     - `flightNumberCounter`: Static counter for flight numbers.
     - `flightNumber`: Unique number for each flight.
     - `destination`: Destination of the flight.
     - `departureTime`: Scheduled departure time.
     - `ArrivalTime`: Scheduled arrival time.
   - **Constructor**:
     - Initialize the attributes and increment the flight number counter.
   - **Methods**:
     - `upcomingFlight()`: Returns true if the departure time is in the future.
     - `isComplete()`: Returns true if the arrival time is in the past.
     - `getFlightNumber()`: Returns the flight number.
     - `updateDeparture(LocalTime)`: Updates the departure time.
     - `updateArrival(LocalTime)`: Updates the arrival time.
     - `display()`: Prints flight details.

2. **Create `Airport` Class**
   - **Attributes**:
     - `name`: Name of the airport.
     - `flights`: List to store Flight objects.
   - **Constructor**:
     - Initialize airport name and create an empty list for flights.
   - **Private Methods**:
     - `findFlightByNumber(int)`: Searches for a flight by its number and returns it if found.
   - **Public Methods**:
     - `addFlight(Flight)`: Adds a flight to the airport's list.
     - `removeFlight(int)`: Removes a flight based on its number and prints a message.
     - `upcomingFlights()`: Returns a list of flights that are yet to depart.

- `CompletedFlights()`: Returns a list of flights that have already arrived.

- `displayDetails()`: Displays all flights at the airport.

Step 2: Main Program Execution

3. **Create Main Class (`airportMain`)**

- In the main method:

1. Create an instance of the `Airport` class with a specified name (e.g., "Demonic Ports").

2. Create several instances of the `Flight` class with different destinations, arrival times, and departure times.

3. Add these flight instances to the airport using the `addFlight()` method.

Step 3: Display Flight Information

4. **Display All Flights**

- Call the `displayDetails()` method on the airport instance to show all flights currently at the airport.

Step 4: Update Flight Information

5. **Update Arrival Time**

- Use the `updateArrival()` method on one of the flight instances to change its arrival time as needed.

- Display updated flight information by calling `displayDetails()` again.

Step 5: Remove a Flight

6. **Remove Flight**

- Call the `removeFlight()` method on the airport instance with a specific flight number to remove it from the list.

- Display remaining flights using `displayDetails()`.

Step 6: List Upcoming and Completed Flights

7. **Display Upcoming Flights**

- Call the `upcomingFlights()` method and display each upcoming flight using their respective display methods.

8. **Display Completed Flights**

   - Call the `CompletedFlights()` method and display each completed flight using their respective display methods.

Conclusion

9. **End of Program**

   - The program concludes after displaying all relevant information about flights at the airport, including updates, removals, upcoming, and completed flights.

Q5:

Code:

```
import java.util.Scanner;
interface Shape {
    double area();
}
class Circle implements Shape {
    double radius;
    Circle(double radius) {
        this.radius = radius;
    }
    public double area() {
        return 3.14159 * radius * radius;
    }
}
class Rectangle implements Shape {
    double length, width;
    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double area() {
        return length * width;
    }
}
class Triangle implements Shape {
    double base, height;
    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
```

```
    public double area() {
        return 0.5 * base * height;
    }
}
public class shapeMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("\nChoose a shape (1: Circle, 2: Rectangle, 3: Triangle, 4: Exit):");
            int choice = sc.nextInt();
            if (choice == 4) {
                System.out.println("Exit");
                break;
            }
            Shape shape = null;
            switch (choice) {
                case 1:
                    System.out.print("Enter radius: ");
                    double radius = sc.nextDouble();
                    shape = new Circle(radius);
                    break;
                case 2:
                    System.out.print("Enter length: ");
                    double length = sc.nextDouble();
                    System.out.print("Enter width: ");
                    double width = sc.nextDouble();
                    shape = new Rectangle(length, width);
                    break;
                case 3:
                    System.out.print("Enter base: ");
                    double base = sc.nextDouble();
```

```
                System.out.print("Enter height: ");
                double height = sc.nextDouble();
                shape = new Triangle(base, height);
                break;
            default:
                System.out.println("Invalid choice.");
                continue;
        }
        System.out.println("The area is: " + shape.area());
    }
    sc.close();
  }
}
```

Output:

```
Choose a shape (1: Circle, 2: Rectangle, 3: Triangle, 4: Exit):
1
Enter radius: 100
The area is: 31415.899999999998

Choose a shape (1: Circle, 2: Rectangle, 3: Triangle, 4: Exit):
2
Enter length: 31
Enter width: 2221
The area is: 68851.0

Choose a shape (1: Circle, 2: Rectangle, 3: Triangle, 4: Exit):
3
Enter base: 22
Enter height: 22.11
The area is: 243.20999999999998

Choose a shape (1: Circle, 2: Rectangle, 3: Triangle, 4: Exit):
4
Exit
PS D:\Mca_upes\codes and stuff\class\MCA_Codings\java\Assignment-1>
```

Algo:

1. Initialize:
    - Create a Scanner object to read user input
    - Set up a while loop to continuously prompt the user for input until they choose to exit
2. Get User Choice:
    - Prompt the user to choose a shape (Circle, Rectangle, Triangle, or Exit)
    - Read the user's choice using the Scanner object
3. Create Shape Object:
    - Based on the user's choice, create a shape object (Circle, Rectangle, or Triangle)
    - Prompt the user to enter the required parameters for the chosen shape
    - Use the entered parameters to create the shape object
4. Calculate Area:
    - Call the area() method on the shape object to calculate its area
5. Display Result:
    - Print the calculated area to the console
6. Repeat or Exit:
    - If the user chose to exit, break out of the loop
    - Otherwise, continue to the next iteration of the loop