```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Patolli {

	static Map<Integer, Position> positionMap = new HashMap<Integer, Patolli.Position>();

	public static void main(String[] args) throws FileNotFoundException {
		Patolli patolli = new Patolli();
		positionMap = patolli.initialize();
		File text = new File("C:\\eclipse-workspace\\MyJavaProj\\src\\input.txt");
		// Creating Scanner instnace to read File in Java
		Scanner scanner = new Scanner(text);

		String line;
		String arr[];
		// Reading each line of file using Scanner class
		while (scanner.hasNextLine()) {
			line = scanner.nextLine();
			arr = line.split(" ");
			patolli.setOpponentMarkers(arr);
			int result = patolli.move(arr);
			if (result == 52) {
				System.out.println(" Input : " + line + " Out Put : GAME OVER");
			} else {
				System.out.println(" Input : " + line + " Out Put : " + result);
```

```
                }
            }

        }

        private int move(String arr[]) {
            int currentPos = Integer.valueOf(arr[3]);
            int noOfRolls = Integer.valueOf(arr[4]);
            int diceRoll;
            Position pos;
            int tempPosition;
            for (int i = 5; i < arr.length; i++) {
                diceRoll = Integer.valueOf(arr[i]);
                tempPosition = currentPos;
                if (currentPos + diceRoll > 52) {
                    currentPos = tempPosition;
                } else {
                    pos = positionMap.get(currentPos + diceRoll);
                    if (!pos.isOccupied()) {
                        if (pos.isPrime()) {
                            currentPos = getForwardPosition(pos.getPositionNo(),
pos.getPositionNo() + 6, 1);
                        } else if (pos.isPerfSquare()) {
                            currentPos = getBackwardPosition(pos.getPositionNo(),
pos.getPositionNo() - 6, -1);
                        } else if (isHandV(currentPos, diceRoll)) {
                            // Handle H and V
                            // currentPos = getPosition(currentPos, 52, 5);
                            int rem = 0;
```

```
                                        rem = currentPos % diceRoll;
                                        currentPos = currentPos + diceRoll - rem;
                                        if (positionMap.get(currentPos).isOccupied()) {
                                                currentPos = tempPosition;
                                        }
                                        // }while (positionMap.get(currentPos).isOccupied() );
                                } else {
                                        currentPos = pos.getPositionNo();
                                }
                                if (currentPos > 52) {
                                        currentPos = tempPosition;
                                }
                        }
                        // System.out.println("Dice Roll : "+diceRoll+" Position: "+currentPos);
                }
        }
        return currentPos;
}

private boolean isHandV(int currentPos, int diceRoll) {
        int hvArr[] = { 7, 12, 17, 22, 27, 35, 40, 45, 50 };
        for (int pos : hvArr) {
                if (currentPos < pos && pos < currentPos + diceRoll) {
                        return true;
                }

        }
        return false;
```

```
}

private int getForwardPosition(int start, int end, int increment) {
	for (int i = start; i <= end; i += increment) {
		if (positionMap.get(i).isOccupied()) {
			return i - 1;
		}
	}
	return end;
}

private int getBackwardPosition(int start, int end, int increment) {
	for (int i = start; i >= end; i += increment) {
		if (positionMap.get(i).isOccupied()) {
			return i + 1;
		}
	}
	return end;
}

private void setOpponentMarkers(String arr[]) {

	for (int i = 1; i <= positionMap.size(); i++) {
		positionMap.get(i).setOccupied((false));
	}
	for (int i = 0; i < 3; i++) {
		positionMap.get(Integer.valueOf(arr[i])).setOccupied((true));
	}
```

```
}

private Map<Integer, Position> initialize() {

	Map<Integer, Position> retMap = new HashMap<Integer, Patolli.Position>();
	Position pos;
	for (int i = 1; i < 53; i++) {
		pos = new Position(i, false);
		retMap.put(i, pos);
	}
	return retMap;

}

class Position {

	int positionNo;
	boolean isOccupied = false;

	Position(int positionNo, boolean isOccupied) {
		this.positionNo = positionNo;
		this.isOccupied = isOccupied;
	}

	public int getPositionNo() {
		return positionNo;
	}

	public void setPositionNo(int positionNo) {
```

```
        this.positionNo = positionNo;
}

public boolean isOccupied() {
        return isOccupied;
}

public void setOccupied(boolean isOccupied) {
        this.isOccupied = isOccupied;
}

public boolean isPrime() {
        if (positionNo <= 1) {
                return false;
        }
        for (int i = 2; i <= Math.sqrt(positionNo); i++) {
                if (positionNo % i == 0) {
                        return false;
                }
        }
        return true;
}

public boolean isPerfSquare() {
        if (positionNo <= 4) {
                return false;
        }
        double sq = Math.sqrt(positionNo);
        return ((sq - Math.floor(sq)) == 0);
```

```
                }

        }

}
```