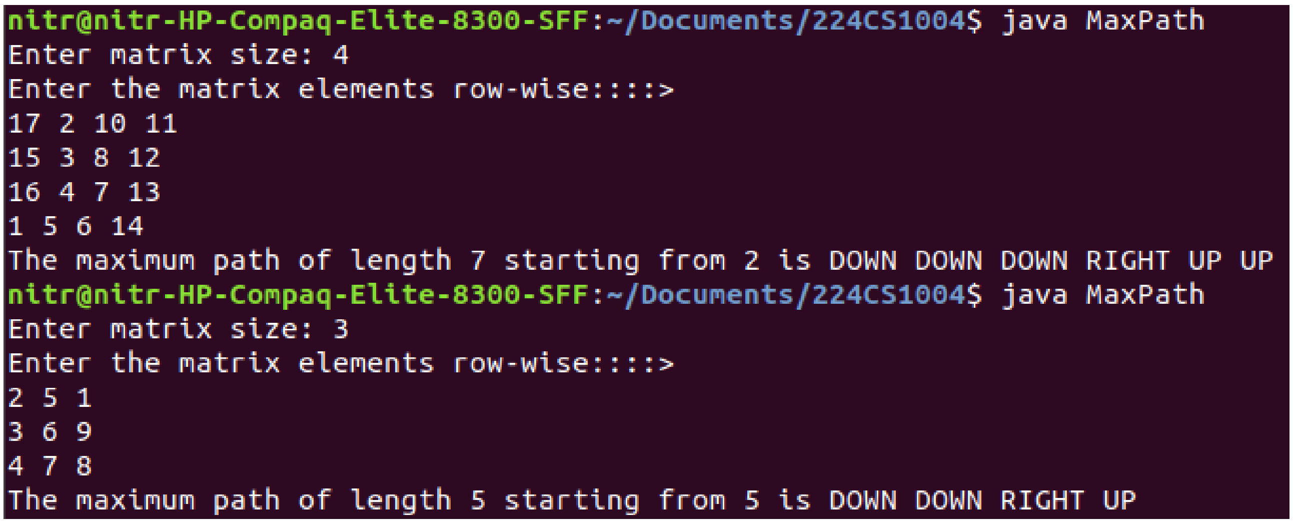
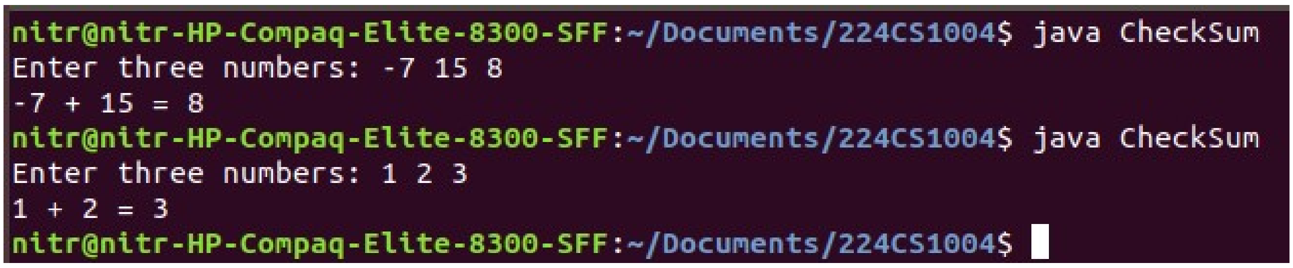


```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$ java MaxPath
Enter matrix size: 4
Enter the matrix elements row-wise::::>
17 2 10 11
15 3 8 12
16 4 7 13
1 5 6 14
The maximum path of length 7 starting from 2 is DOWN DOWN DOWN RIGHT UP UP
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$ java MaxPath
Enter matrix size: 3
Enter the matrix elements row-wise::::>
2 5 1
3 6 9
4 7 8
The maximum path of length 5 starting from 5 is DOWN DOWN RIGHT UP
```

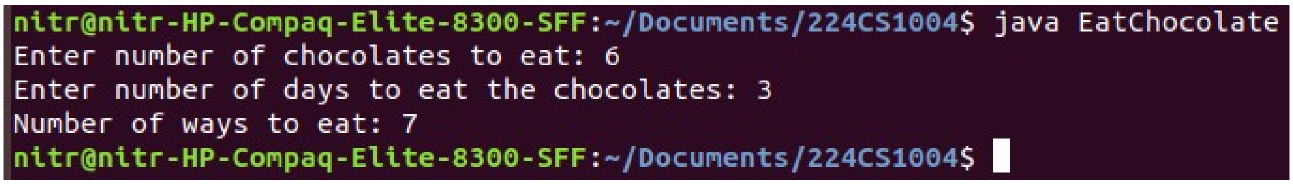
```
import java.io.*;
import java.util.Scanner;

/**
* Object to hold the maxPath details
*/
class Path {

	int srcX, srcY, length, currX, currY;
	String path;

	public Path(int srcX, int srcY, int length, String path) {
		this.srcX = srcX; this.srcY = srcY;
		this.length = length;
		this.path = path;
		this.currX = srcX; this.currY = srcY;
	}

	/**
	* Moves the path along according to current move
	*/
	public void movePath(String move) {
		switch (move.charAt(0)) {
			case 'U': this.currX--; break;
			case 'D': this.currX++; break;
			case 'L': this.currY--; break;
			case 'R': this.currY++; break;
		}

		this.length++;
		this.path += move; // adding path move
	}

	/**
	* Copies path data
	*/
	public void copy(Path path) {
		this.srcX = path.srcX; this.srcY = path.srcY;
		this.length = path.length;
		this.path = path.path;
		this.currX = path.srcX; this.currY = path.srcY;
	}

}

public class MaxPath {

	/**
	* Move to the correct cell and send back the move
	* "" if none is valid
	*/
	public static String getMove(int[][] matrix, int x, int y) {
		int n = matrix.length;

		String move = "";
		if (x-1 >= 0 && matrix[x][y]+1 == matrix[x-1][y]) move = "UP ";
		else if (x+1 < n && matrix[x][y]+1 == matrix[x+1][y]) move = "DOWN ";
		else if (y-1 >= 0 && matrix[x][y]+1 == matrix[x][y-1]) move = "LEFT ";
		else if (y+1 < n && matrix[x][y]+1 == matrix[x][y+1]) move = "RIGHT ";

		return move;
	}


	/**
	* This function updates the maxPath based on the current provided Path
	*/
	public static void updatePath(int[][] matrix, boolean[][] visited, Path currPath, Path
maxPath) {

		// current node
		int x = currPath.currX, y = currPath.currY;
		visited[x][y] = true;

		// check which move is valid from current node and update currPath
		String move = getMove(matrix, x, y);

		if (!move.equals("")) {
			currPath.movePath(move); // move the current path based on move

			if (maxPath.length < currPath.length) {
				maxPath.copy(currPath);
			}
			updatePath(matrix, visited, currPath, maxPath); // update the path again
		} else {
			return;
		}
```

```
            }


    /**
    * Main driver code
    */
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter matrix size: ");
        int n = sc.nextInt();
        sc.nextLine(); // reading till line end to avoid errors

        // matrix init
        int[][] matrix = new int[n][n];
        boolean[][] visited = new boolean[n][n];

        // input of matrix
        System.out.println("Enter the matrix elements row-wise::::>");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = sc.nextInt();
            }
            sc.nextLine();
        }

        Path maxPath = new Path(-1, -1, 0, ""); // to hold the current maxPath

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                    if (!visited[i][j]) {
                            Path currPath = new Path(i, j, 1, ""); // creating a temporary path for holding
current values
                            updatePath(matrix, visited, currPath, maxPath); // update the maxPath
keeping the current node as source
                    }
            }
        }

        System.out.printf("The maximum path of length %d starting from %d is %s\n",
maxPath.length, matrix[maxPath.srcX][maxPath.srcY], maxPath.path);

        sc.close();
    }

}
```



```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$ java CheckSum
Enter three numbers: -7 15 8
-7 + 15 = 8
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$ java CheckSum
Enter three numbers: 1 2 3
1 + 2 = 3
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$
```

```java
import java.io.*;
import java.util.Scanner;

public class CheckSum {

    /**
    * Main driver code
    */
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter three numbers: ");
        int a = sc.nextInt(), b = sc.nextInt(), c = sc.nextInt();
        sc.nextLine(); // reading till line end to avoid errors

        if (a+b == c) System.out.printf("%d + %d = %d\n", a, b, c);
        if (b+c == a) System.out.printf("%d + %d = %d\n", b, a, c);
        if (a+c == b) System.out.printf("%d + %d = %d\n", a, c, b);

        sc.close();
    }

}
```


```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$ java EatChocolate
Enter number of chocolates to eat: 6
Enter number of days to eat the chocolates: 3
Number of ways to eat: 7
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Documents/224CS1004$
```

```
import java.io.*;
import java.util.Scanner;

public class EatChocolate {

        /**
        * Eat chocolate function
        */
        public static int eat(int N, int k, int count) {
                if (k == 0) {
                        if (N == 0) return 1;
                        else if (N > 0 || N < 0) return 0;
                }
                return eat(N-1, k-1, count) + eat(N-2, k-1, count) + eat(N-3, k-1, count);
        }

   /**
   * Main driver code
   */
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of chocolates to eat: ");
      int N = sc.nextInt(); sc.nextLine();

      System.out.print("Enter number of days to eat the chocolates: ");
      int k = sc.nextInt(); sc.nextLine();

      int count = 0; // count of ways

      count = eat(N-1, k-1, count) + eat(N-2, k-1, count) + eat(N-3, k-1, count);

                System.out.println("Number of ways to eat: " + count);

      sc.close();
   }

}
```