```
//Nipun
//Intermediate -Patolli

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class NipunJ_INTc4Patolli {
    private static int[][] matrix = { { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0,
    1, 52, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 2, 51, 0, 0, 0, 0, 0 }, { 0, 7, 6, 5, 4, 3, 50, 49, 48, 47, 46, 0 },
            { 0, 8, 9, 10, 11, 12, 41, 42, 43, 44, 45, 0 }, { 0, 17, 16, 15, 14, 13, 40, 39,
            38, 37, 36, 0 },
            { 0, 18, 19, 20, 21, 22, 31, 32, 33, 34, 35, 0 }, { 0, 0, 0, 0, 0, 23, 30, 0, 0, 0,
            0, 0 },
            { 0, 0, 0, 0, 0, 24, 29, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 25, 28, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 26, 27, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } };
    private final static int[][] finalMatrix = { { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 1, 52, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 2, 51, 0, 0, 0, 0, 0 },
            { 0, 7, 6, 5, 4, 3, 50, 49, 48, 47, 46, 0 }, { 0, 8, 9, 10, 11, 12, 41, 42, 43, 44,
            45, 0 },
            { 0, 17, 16, 15, 14, 13, 40, 39, 38, 37, 36, 0 }, { 0, 18, 19, 20, 21, 22, 31, 32,
            33, 34, 35, 0 },
            { 0, 0, 0, 0, 0, 23, 30, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 24, 29, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 25, 28, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 26, 27, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } };

    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(new File("patolli"));
        while (sc.hasNextLine()) {
            for (int i = 0; i < finalMatrix.length; i++) {
                for (int j = 0; j < finalMatrix[0].length; j++) {
                    matrix[i][j] = finalMatrix[i][j];
                }
            }
            String input = sc.nextLine();
            String[] inputArray = input.split(" ");
            fillOpps(Integer.parseInt(inputArray[0]));
            fillOpps(Integer.parseInt(inputArray[1]));
            fillOpps(Integer.parseInt(inputArray[2]));
            fillPlayers(Integer.parseInt(inputArray[3]));
            fillPlayers(Integer.parseInt(inputArray[4]));
            fillPlayers(Integer.parseInt(inputArray[5]));
//          for (int i = 0; i < matrix.length; i++) {
//              for (int j = 0; j < matrix[0].length; j++) {
//                  if (matrix[i][j] != 0)
//                      System.out.format("%4d", matrix[i][j]);
//                  else
//                      System.out.format("%4s", "");
//              }
//              System.out.println();
//          }
            for (int i = 7; i < 7 + Integer.parseInt(inputArray[6]); i++) {
                goNext(Integer.parseInt(inputArray[i]));
//              System.out.println("\n");
//              for (int c = 0; c < matrix.length; c++) {
//                  for (int j = 0; j < matrix[0].length; j++) {
//                      if (matrix[c][j] != 0)
//                          System.out.format("%4d", matrix[c][j]);
//                      else
//                          System.out.format("%4s", "");
//                  }
//                  System.out.println();
//              }
            }

            getPlaces();
```

```
    }
}

public static void fillPlayers(int n) {
    for (int i = 1; i < matrix.length; i++) {
        for (int j = 1; j < matrix.length; j++) {
            if (matrix[i][j] == n) {
                matrix[i][j] = -1;
            }
        }
    }
}

public static void fillOpps(int n) {
    for (int i = 1; i < matrix.length; i++) {
        for (int j = 1; j < matrix.length; j++) {
            if (matrix[i][j] == n) {
                matrix[i][j] = -2;
            }
        }
    }
}

public static void goNext(int n) {
    ArrayList<Integer> horList = new ArrayList<Integer>();
    int i = 1;
    int j = 5;
    int tempI = 0;
    int tempJ = 0;
    while (matrix[i][j] != -1) {
        if (finalMatrix[i + 1][j] == finalMatrix[i][j] + 1) {
            i++;
        } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] + 1) {
            i--;
        } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] + 1) {
            j++;
        } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] + 1) {
            j--;
        }
    }
    int startI = i;
    int startJ = j;
    tempI = i;
    tempJ = j;
    matrix[i][j] = finalMatrix[i][j];
    for (int k = 0; k < n; k++) {
        if (finalMatrix[i + 1][j] == finalMatrix[i][j] + 1) {
            i++;
            horList.add(0);
        } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] + 1) {
            i--;
            horList.add(0);
        } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] + 1) {
            j++;
            horList.add(1);
        } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] + 1) {
            j--;
            horList.add(1);
        }
    }
    if (matrix[i][j] < 0) {
        i = tempI;
        j = tempJ;
        horList.clear();
        horList.add(0);
    }
    matrix[i][j] = -1;
    int b = 0;
    if (isPrime(finalMatrix[i][j])) {
```

```
            matrix[i][j] = finalMatrix[i][j];
            while (matrix[i][j] > 0 && b < 6) {
                tempI = i;
                tempJ = j;
                if (finalMatrix[i + 1][j] == finalMatrix[i][j] + 1) {
                    i++;
                } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] + 1) {
                    i--;
                } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] + 1) {
                    j++;
                } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] + 1) {
                    j--;
                }
                b++;
            }
            if (matrix[i][j] < 0) {
                i = tempI;
                j = tempJ;
            }
            matrix[i][j] = -1;
        } else if (isPerfect(finalMatrix[i][j]) && finalMatrix[i][j] > 4) {
            matrix[i][j] = finalMatrix[i][j];
            while (matrix[i][j] > 0 && b < 6) {
                tempI = i;
                tempJ = j;
                if (finalMatrix[i + 1][j] == finalMatrix[i][j] - 1) {
                    i++;
                } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] - 1) {
                    i--;
                } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] - 1) {
                    j++;
                } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] - 1) {
                    j--;
                }
                b++;
            }
            if (matrix[i][j] < 0) {
                i = tempI;
                j = tempJ;
            }
            matrix[i][j] = -1;
        } else if (horList.get(0) == 1 && horList.contains(0)) {
            tempI = i;
            tempJ = j;
            matrix[i][j] = finalMatrix[i][j];
            matrix[startI][startJ] = finalMatrix[startI][startJ];
            for (int t = 0; t < n; t++) {
                if ((finalMatrix[i][j] % n != 0) && (matrix[i][j] > 0)) {
                    if (finalMatrix[i + 1][j] == finalMatrix[i][j] - 1) {
                        i++;
                    } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] - 1) {
                        i--;
                    } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] - 1) {
                        j++;
                    } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] - 1) {
                        j--;
                    }
                } else if (matrix[i][j] < 0) {
                    if (finalMatrix[i + 1][j] == finalMatrix[i][j] - 1) {
                        i++;
                    } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] - 1) {
                        i--;
                    } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] - 1) {
                        j++;
                    } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] - 1) {
                        j--;
                    }
                } else if ((finalMatrix[i][j] % n == 0) && (matrix[i][j] > 0)) {
//                  matrix[startI][startJ] = finalMatrix[startI][startJ];
```

```
                    matrix[i][j] = -1;
                    break;
                } else if (t == n - 1) {
                    i = startI;
                    j = startJ;
                    break;
                }
            }
            matrix[tempI][tempJ] = finalMatrix[tempI][tempJ];
            matrix[i][j] = -1;
        }
        if (matrix[1][6] == -1) {
            matrix[1][6] = finalMatrix[1][6];
        }
    }

    public static boolean isPrime(int n) {
        if (n <= 1) {
            return false;
        }
        for (int i = 2; i < n; i++) {
            if (n % i == 0)
                return false;
        }
        return true;
    }

    public static boolean isPerfect(int n) {
        for (int i = 0; i < n; i++) {
            if (i * i == n) {
                return true;
            }
        }
        return false;
    }

    public static void getPlaces() {
        int i = 1;
        int j = 5;
        int c = 0;
        while (c < 3 && finalMatrix[i][j] < 51) {
            if (finalMatrix[i + 1][j] == finalMatrix[i][j] + 1) {
                i++;
            } else if (finalMatrix[i - 1][j] == finalMatrix[i][j] + 1) {
                i--;
            } else if (finalMatrix[i][j + 1] == finalMatrix[i][j] + 1) {
                j++;
            } else if (finalMatrix[i][j - 1] == finalMatrix[i][j] + 1) {
                j--;
            }
            if (matrix[i][j] == -1) {
                System.out.print(finalMatrix[i][j] + " ");
                c++;
            }
        }
        System.out.println();
    }
}
```