```
//ShreyaAravindan
//Intermediate -Patolli

import java.util.*;
public class ShreyaAravindan_INTc4Patolli{
   public static boolean moved = false;
   public static boolean happ = false;
   public static int position = 0;

   public static void main(String[] args) {
          for(int i = 0; i<5; i++) {
          game();
         }
   }
   public static void answer(ArrayList<Integer> mainP) {
      for (int i = 0; i < mainP.size(); i++) {
         System.out.print(mainP.get(i) + " ");
      }
      System.out.println();
   }
   public static Boolean isSquare(int num, int[]gboard){
      for (int i = 0; i < num / 2 + 2; i++){
         if (i * i == num){
            if(num<gboard.length) {
               return true;
            }
         }
      }
      return false;
   }
   public static boolean isPrime(int num){
      if(num<=1)
         return false;
      for(int i = 2; i<num; i++)
         if(num%i ==0)
            return false;
      return true;
   }
   public static boolean ifOccupied(int num,int[]gboard){
          if (gboard[num] != 0)
             return true;
          return false;
       }
   public static void game() {
       ArrayList<Integer> Opponent = new ArrayList<Integer>();
       ArrayList<Integer> mainP = new ArrayList<Integer>();
          Scanner s = new Scanner(System.in);
          System.out.println("Enter Line: ");
          String str = s.nextLine();
          String[]arr = str.split(" ");
          int[]arr2 = new int[arr.length];
          for(int i = 0; i<arr.length; i++){
             arr2[i]= Integer.parseInt(arr[i]);
          }
          List<Integer>arrList = new ArrayList<Integer>();
          for(int a : arr2){
             arrList.add(Integer.valueOf(a));
          }
          for(int i = 0; i<3;i++){
             Opponent.add(arrList.get(i));
          }
          for(int i = 3; i<6; i++){
             mainP.add(arrList.get(i));
          }
          Collections.sort(mainP);
          int NumofRolls = arrList.get(6);
          ArrayList<Integer> numbers = new ArrayList<Integer>();
          for(int i = 7; i<arrList.size();i++) {
              numbers.add(arrList.get(i));
```

```
 }
int gboard[] = new int[52];
for (int i = 0; i < gboard.length; i++) {
   gboard[i] = 0;
}
for (int i = 0; i < 3; i++) {
   gboard[Opponent.get(i)-1] = 2;
}
int gboard2[] = new int[gboard.length];
for (int i = 0; i < gboard.length; i++) {
   gboard2[i] = gboard[i];
}
for (int y = 0; y < NumofRolls; y++) {
   for (int i = 0; i < gboard.length; i++) {
      gboard[i] = gboard2[i];
   }
   for (int i = 0; i < mainP.size(); i++) {
      gboard[mainP.get(i)-1] = 1;
   }
   int rollN = numbers.get(y);
   int oPos = mainP.get(0);
   mainP.remove(0);
   int c = oPos;
   if (ifOccupied(oPos + rollN - 1, gboard)==false) {
      oPos = oPos+ rollN;
   }
   else {
      mainP.add(oPos);
      Collections.sort(mainP);
      continue;
   }
   int num = 1;
   int count = 0;
   happ = false;
   while (num > 0 && count < 20) {
      count++;
      num = 0;
      if (oPos == gboard.length) {
         gboard[51] = 0;
      }
      if (isPrime(oPos)==true) {
         happ = true;
         if (oPos + 6 <= gboard.length) {
            int count2 = 1;
            moved = false;
            while (gboard[oPos + count2-1] == 0 && count2 < 6) {
               count2++;
               moved = true;
            }
            if (moved==true) {
               oPos = oPos+ count2 - 1;
               num++;
            }
         }
      }
      if (isSquare(oPos, gboard)== true) {
         happ = true;
         int count2 = 1;
         moved = false;
         while (gboard[oPos-count2-1] == 0 && count2 < 6) {
            count2++;
            moved = true;
         }
         if (moved==true) {
            count2 = count2-1;
            oPos= oPos - count2;
            num++;
         }
      }
```

```
            if (happ==false) {
               int board[] = {7, 12, 17, 22, 27, 35, 40, 45, 50};
               oPos = oPos- rollN;
               if (corners(c, oPos + rollN, board)==true) {
                  int pos2 = board[position];
                  oPos = ninth(pos2, rollN, oPos,gboard);
                  continue;
               }
               oPos = oPos+ rollN;
            }
         }
         if(oPos<52) {
         mainP.add(oPos);
         }
         Collections.sort(mainP);
      }
      answer(mainP);
   }
   public static boolean corners(int c, int num,int[]board){
      for (int i=0; i<9; i++){
         if (c < board[i] && num > board[i]) {
            position = i;
            return true;
         }
      }
      return false;
   }
   public static int ninth(int pos2, int rollN, int oPos,int[]gboard){
      for (int i=pos2; i<=oPos+rollN; i++){
         if (i%rollN == 0 && gboard[i-1] == 0)
            return i;
      }
      return oPos;
   }
}
```