题目描述：

有 N（3<=N<10000）个运动员，他们的 id 为 0 到 N-1,他们的实力由一组整数表示。他们之间进行比赛，需要决出冠亚军。比赛的规则是 0 号和 1 号比赛，2 号和 3 号比赛，以此类推，每一轮，相邻的运动员进行比赛，获胜的进入下一轮;实力值大的获胜，实力值相等的情况，id 小的情况下获胜;,轮空的直接进入下一轮.

输入描述：

输入一行 N 个数字代表 N 的运动员的实力值(0<=实力值<=10000000000)。

输出描述：

输出冠亚军的 id，用空格隔开。

示例 1

输入：

```
2 3 4 5
```

输出：

```
3 1 2
```

说明：

第一轮比赛，id 为 0 实力值为 2 的运动员和 id 为 1 实力值为 3 的运动员比赛， 1 号胜出进入下一轮争夺冠亚军，id 为 2 的运动员和 id 为 3 的运动员比赛，3 号胜出进入下一轮争夺冠亚军；冠亚军比赛，3 号胜 1 号；故冠军为 3 号， 亚军为 1 号。2 号与 0 号，比赛进行季军的争夺，2 号实力值为 4，0 号实力值 2，故 2 号胜出，得季军。冠亚季军为 3 1 2。

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
```

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] scoreList = sc.nextLine().split(" ");
        List<Player> players = new ArrayList<>();
        for (int i = 0; i < scoreList.length; i++) {
                players.add(new Player(i, Long.parseLong(scoreList[i])));
        }
        List<Player> res = play(players);
        System.out.println(res.get(0).index + " " + res.get(1).index + " " + res.get(2).index);
}

private static List<Player> play(List<Player> players) {
        List<Player> res = new ArrayList<>();
        if (players.size() == 4) {
                Player loss1 = null;
                Player win1 = null;
                if (players.get(0).score >= players.get(1).score) {
                        win1 = players.get(0);
                        loss1 = players.get(1);
                } else {
                        win1 = players.get(1);
                        loss1 = players.get(0);
                }

                Player loss2 = null;
                Player win2 = null;
                if (players.get(2).score >= players.get(3).score) {
                        win2 = players.get(2);
                        loss2 = players.get(3);
                } else {
                        win2 = players.get(3);
                        loss2 = players.get(2);
                }

                if (win1.score >= win2.score) {
                        res.add(win1);
                        res.add(win2);
                } else {
                        res.add(win2);
                        res.add(win1);
                }

                if (loss1.score >= loss2.score) {
                        res.add(loss1);
```

```
        } else {
            res.add(loss2);
        }

        return res;
    }
//三人
    if (players.size() <= 3) {
        Player loss = null;
        Player win = null;
        if (players.get(0).score >= players.get(1).score) {
            loss = players.get(1);
            win = players.get(0);
        } else {
            loss = players.get(0);
            win = players.get(1);
        }
        if (win.score >= players.get(2).score) {
            res.add(win);
            res.add(players.get(2));
        } else {
            res.add(players.get(2));
            res.add(win);
        }
        players.add(loss);
        return players;
    }

    int playSize = players.size() % 2 == 1 ? players.size() / 2 + 1 : players.size() / 2;

    for (int i = 0; i < playSize; i++) {
        if (2 * i + 1 >= players.size()) {
            res.add(players.get(2 * i));
        } else {
            Player player1 = players.get(2 * i);
            Player player2 = players.get(2 * i + 1);

            if (player1.score >= player2.score) {
                res.add(player1);
            } else {
                res.add(player2);
            }
        }
    }
```

```
            return play(res);
        }


        static class Player {

            int index;
            long score;

            public Player(int index, long score) {
                this.index = index;
                this.score = score;
            }
        }
}
```