using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace \_02\_ManipulateArrays

{

public class ManipulateArrays

{

public static void Main()

{

var inputArr = Console.ReadLine().Split(' ').Select(int.Parse).ToList();

var command = Console.ReadLine();

while (command != "end")

{

var splitCommand = command.Split(' ').ToArray();

if (splitCommand[0] == "exchange")

{

var index = int.Parse(splitCommand[1]);

if (index < 0 || index >= inputArr.Count)

{

Console.WriteLine("Invalid index");

}

else if (index != inputArr.Count - 1)

{

inputArr = ExchangeArray(index, inputArr);

}

}

else if (splitCommand[0] == "max")

{

var index = FindIndexOfMax(splitCommand[1], inputArr);

if (index == -1)

{

Console.WriteLine("No matches");

}

else

{

Console.WriteLine(index);

}

}

else if (splitCommand[0] == "min")

{

var index = FindIndexOfMin(splitCommand[1], inputArr);

if (index == -1)

{

Console.WriteLine("No matches");

}

else

{

Console.WriteLine(index);

}

}

else if (splitCommand[0] == "first")

{

var countElmnts = int.Parse(splitCommand[1]);

if (countElmnts > inputArr.Count)

{

Console.WriteLine("Invalid count");

}

else

{

var firstElemntsArr = new List<int>();

var oddOrEven = splitCommand[2];

firstElemntsArr = ReturnFirst(inputArr, countElmnts, oddOrEven);

Console.WriteLine($"[{String.Join(", ", firstElemntsArr)}]");

}

}

else if (splitCommand[0] == "last")

{

var countElmnts = int.Parse(splitCommand[1]);

if (countElmnts > inputArr.Count)

{

Console.WriteLine("Invalid count");

}

else

{

var lastElemntsArr = new List<int>();

var oddOrEven = splitCommand[2];

lastElemntsArr = ReturnLast(inputArr, countElmnts, oddOrEven);

Console.WriteLine($"[{String.Join(", ", lastElemntsArr)}]");

}

}

command = Console.ReadLine();

}

Console.WriteLine($"[{String.Join(", ", inputArr)}]");

}

public static List<int> ReturnFirst(List<int> inputArr, int countElmnts, string oddOrEven)

{

if (oddOrEven == "odd")

{

return inputArr.Where(i => i % 2 == 1).Take(countElmnts).ToList();

}

else

{

return inputArr.Where(i => i % 2 == 0).Take(countElmnts).ToList();

}

}

public static List<int> ReturnLast(List<int> inputArr, int countElmnts, string oddOrEven)

{

if (oddOrEven == "odd")

{

return inputArr.Where(i => i % 2 == 1)

.Reverse().Take(countElmnts).Reverse().ToList();

}

else

{

return inputArr.Where(i => i % 2 == 0)

.Reverse().Take(countElmnts).Reverse().ToList();

}

}

public static int FindIndexOfMax(string v, List<int> inputArr)

{

var indexOfMax = -1;

if (v == "odd")

{

var odds = inputArr.Where(i => i % 2 == 1).ToList();

if (odds.Count > 0)

{

indexOfMax = inputArr.LastIndexOf(odds.Max());

}

}

else if (v == "even")

{

var even = inputArr.Where(i => i % 2 == 0).ToList();

if (even.Count > 0)

{

indexOfMax = inputArr.LastIndexOf(even.Max());

}

}

return indexOfMax;

}

public static int FindIndexOfMin(string v, List<int> inputArr)

{

var indexOfMin = -1;

if (v == "odd")

{

var odds = inputArr.Where(i => i % 2 == 1).ToList();

if (odds.Count > 0)

{

indexOfMin = inputArr.LastIndexOf(odds.Min());

}

}

else if (v == "even")

{

var even = inputArr.Where(i => i % 2 == 0).ToList();

if (even.Count > 0)

{

indexOfMin = inputArr.LastIndexOf(even.Min());

}

}

return indexOfMin;

}

public static List<int> ExchangeArray(int index, List<int> inputArr)

{

var lastPart = inputArr.Skip(index + 1).ToList();

inputArr = inputArr.Take(index + 1).ToList();

inputArr.InsertRange(0, lastPart);

return inputArr;

}

}

}