Tìm phần tử lớn nhất và nhỏ nhất trong Linked List

Khai báo và hiện thực phương thức (method) tìm phần tử lớn nhất và nhỏ nhất trong danh sách liên kết. Tên phương thức phải là GetMaxMinIndex, khai báo thích hợp vào trong chương trình mẫu trên Spoj (/EIUDISC2/problems/ EIULINKED3/), không thay đổi code trong bất kỳ phần nào khác

Gợi ý: Chương trình và cấu trúc dữ liệu tương tự bài EILINKEA - Linked List (Simple Version).

Input

Dòng đầu tiên gồm hai số n, m ($0 \le n \le 2*10^5$, $0 \le m \le 10^5$), là số lượng phần tử ban đầu của danh sách, và số lượng câu lệnh (command)

Dòng tiếp theo chứa n phần tử a_i ban đầu của danh sách ($a_i \le 10^9$)

m dòng tiếp theo chứa các câu lệnh theo theo 1 trong hai dạng:

- "insertAt 0 number": chèn số number vào đầu danh sách liên kết ($|number| \le 10^{9)}$
- "getmaxmin": tìm chỉ số của số lớn nhất và số nhỏ nhất trong danh sách liên kết hiện tại.

Lưu ý: "insertAt 0 number" đã được hiện thực trong chương trình mẫu

Output

Với các command getmaxmin, xuất chỉ số của phần tử lớn nhất và nhỏ nhất tại thời điểm tương ứng. Nếu có nhiều phần tử lớn nhất hoặc nhỏ nhất, xuất ra chỉ số đầu tiên.

Nếu không tồn tại số lớn nhất (số nhỏ nhất) thì xuất ra -1 ở vị trí tương ứng.

Example

Input:
2 4
5 9
insertAt 0 -1
getmaxmin
insertAt 0 9
getmaxmin

Output:
2 0
0 1

C# code

```
using System;
```

```
using System.Text;

namespace Final
{
    class EIULINKED3
    {
        static void Main(string[] args)
        {
            int n = NextInt(), m = NextInt();

            int[] initNumbers = new int[n];
            for (var i = 0; i < n; i++)
            {
                initNumbers[i] = NextInt();
            }

            LinkedList linkedList = new LinkedList();
            for (var i = initNumbers.Length - 1; i >= 0; i--)
            {
                linkedList.InsertAt(0, initNumbers[i]);
            }

            StringBuilder outBf = new StringBuilder();
            for (int i = 0; i < m; i++)
            {
                var command = Next();
                if (command == "insertAt")
                {
                    var index = NextInt();
                    var number = NextInt();
                    linkedList.InsertAt(index, number);
                }
                else if (command == "getmaxmin")
                {
                    var maxMin = linkedList.GetMaxMinIndex();
                    outBf.Append(maxMin.maxIndex + " " + maxMin.minIndex + "\n");
                }
            }
            Console.Write(outBf);
        }

        #region Input Wrapper
        static int s_index = 0;
        static string[] s_tokens;

        private static string Next()
        {
            while (s_tokens == null || s_index == s_tokens.Length)
            {
                s_tokens = Console.ReadLine().Split(new char[] { ' ', '\t', '\n' }, StringSplitOptions.RemoveEmptyEntries);
                s_index = 0;
            }
            return s_tokens[s_index++];
        }

        private static int NextInt()
        {
            return Int32.Parse(Next());
        }
```

```
        #endregion
    }

    class LinkedList
    {
        class LinkedNode
        {
            public int Item { get; set; }
            public LinkedNode Next { get; set; }

            public LinkedNode(int item)
            {
                Item = item;
            }
        }

        LinkedNode head = null;

        public void InsertAt(int index, int number)
        {
            var node = new LinkedNode(number);
            node.Next = head;
            head = node;
        }

        public MaxMinResult GetMaxMinIndex()
        {
            // Your code start here
        }

    }

    class MaxMinResult
    {
        public int maxIndex = 0;
        public int minIndex = 0;
    }
}
```