МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет: Информатика и вычислительная техника

Кафедра: КБИС

Лабораторная работа №6

Деревья: двоичное дерево поиска, обходы дерева,

дерево отрезков, декартово дерево

Выполнил – ст. гр. ВКБ33 Новиков В.Д.

Ростов-на-Дону

Задача №2791. В начало строя!

Капрал Питуца любит командовать своим отрядом. Его любимый приказ «в начало строя». Он выстраивает свой отряд в шеренгу и оглашает последовательность приказов. Каждая приказ имеет вид «Солдаты с l_i по r_i — в начало строя!»

Пронумеруем солдат в начальном положении с 1 до n, слева направо. Приказ «Солдаты с l_i по r_i — в начало строя!» означает, что солдаты, стоящие с l_i по r_i включительно, перемещаются в начало строя, сохраняя относительный порядок.

Например, если в некоторый момент солдаты стоят в порядке $2, 3, 6, 1, 5, 4$, после приказа: «Солдаты с 2 по 4 — в начало строя!» порядок будет $3, 6, 1, 2, 5, 4$.

По данной последовательности приказов найти конечный порядок солдат в строю.

Входные данные
В первой строке два целых числа n and m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — количество солдат и количество приказов. Следующие m строк содержат по два целых числа l_i и r_i ($1 \le l_i \le r_i \le n$).

Выходные данные
Выведите n целых чисел — порядок солдат в конечном положении после выполнения всех приказов.

Примеры

входные данные
6 3 2 4 3 5 2 2
выходные данные
1 4 5 2 3 6

ID	Участник	Задача	Дата	Язык	Статус	Пройдено тестов	Баллы	Подробнее
27639090	Владислав Новиков	2791. В начало строя!	2021-09-06 22:44:06	GNU C++ 8.3	OK	30	100	Подробнее

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

bool flag;

//описываем структуру (декартово дерево (типо Бин Дер Поиска ток самобалансирующ.))
struct Item
{
    int cnt, Value, Priority, Summa;
    Item *l, *r;
    Item() { }
    Item(int Priority, int Value) : cnt(0), Value(Value), Priority(Priority), Summa(0), l(), r() { }
};

typedef Item* Pitem;
Pitem Tree, Ta, Tb, Tc;

//Функция PrintTree выводит содержимое декартового дерева.
void PrintTree(Pitem t)
{
    if (!t)  return;
    PrintTree(t->l);
    if (flag) printf(" ");
    printf("%d", t->Value);
    flag = 1;
    PrintTree(t->r);
}

int cnt(Pitem t)
{
    //? в C++ это тернарный оператор, запись выглядит так return t <если true> t->cnt(в t есть переменная cnt) <если false> 0
    return t ? t->cnt : 0;
}

//Функция GetSum возвращает сумму элементов последовательности, хранящуюся в вершинах поддерева с корнем t.
int GetSum(Pitem t)
{
    if (t) return t->Summa;
    return 0;
}

void update(Pitem t)
{
    if (t)
    {
        t->cnt = 1 + cnt(t->l) + cnt(t->r);
        t->Summa = t->Value + GetSum(t->l) + GetSum(t->r);
    }
}

void Merge(Pitem l, Pitem r, Pitem &t)
{
    if (!l || !r) t = l ? l : r;
    else if (l->Priority > r->Priority) Merge(l->r, r, l->r), t = l;
    else Merge(l, r->l, r->l), t = r;
    update(t);
}
```

```
void Split(Pitem t, Pitem &l, Pitem &r, int pos)
{
    if (!t) return void(l = r = 0);
    if (pos <= cnt(t->l)) Split(t->l, l, t->l, pos), r = t;
    else Split(t->r, t->r, r, pos - 1 - cnt(t->l)), l = t;
    update(t);
}

//На позицию pos декартового дерева t вставляем вершину it.
void Insert(Pitem &t, Pitem it, int pos)
{
    Pitem t1, t2;
    Split(t, t1, t2, pos);
    Merge(t1, it, t1);
    Merge(t1, t2, t);
}

int main() {
    //Создадим декартово дерево с неявным ключом. Поскольку изначально солдаты в строю пронумерованы
    //последовательно от 1 до n, то занесем в поле Value вершин дерева значения от 1 до n.
    int n, m;
    scanf("%d %d", &n, &m);
    for (int i = 0; i < n; i++) Insert(Tree, new Item(rand(), i + 1), i);

    //Последовательно выполняем команды капрала. Следует вырезать из декартового дерева
    //отрезок [li, ri] при помощи функции Split и поставить его в начало используя Merge.
    for (int i = 0; i < m; i++)
    {
        int l, r;
        scanf("%d %d", &l, &r);
        Split(Tree, Tb, Tc, r);
        Split(Tb, Ta, Tb, l - 1); // Tree = (Ta, Tb, Tc)
        Merge(Tb, Ta, Tree);   // Ta, Tb, Tc -> Tb, Ta, Tc
        Merge(Tree, Tc, Tree); // Tree = (Tb, Ta, Tc)
    }

    //Выводим порядок солдат в конечном положении после выполнения всех приказов.
    PrintTree(Tree);
    printf("\n");
    system("pause");
    return 0;
}
```

```
D:\Users\Admin\Desktop\учёба\методы программирования\
6 3
2 4
3 5
2 2
1 4 5 2 3 6
Для продолжения нажмите любую клавишу . . .
```