

**Министерство образования Республики Беларусь
Учреждение образования «Полоцкий государственный
университет имени Евфросинии Полоцкой»**

Кафедра технологий
программирования

Алгоритмы и структуры данных
Отчет по лабораторной работе №1

Выполнил

Ланцев Евгений Николаевич.
21-ИТ-1, ФИТ

Проверил

преподаватель
Виноградова А.Д.

Полоцк
2022 г.

Модуль № 2

“Числа Фибоначчи”

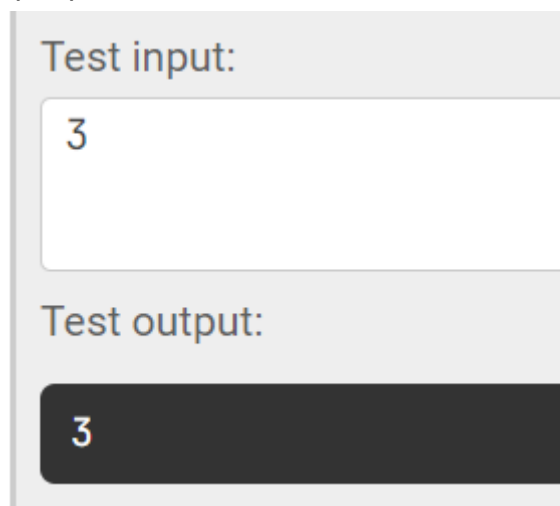
Задача 1: Дано целое число $1 \leq n \leq 40$, необходимо вычислить n -е число Фибоначчи.

Решение

```
class Fibonacci {
public:
    static int get(int n) {
        assert(n >= 0);
        // put your code here
        return n;
    }
};

int main(void) {
    int n;
    std::cin >> n;
    std::cout << Fibonacci::get(n) << std::endl;
    return 0;
}
```

Листинг программы- Подсчет n -ого числа фибоначчи



The image shows a graphical user interface for a program. It has a light gray background. At the top, there is a label "Test input:" in a dark gray font. Below it is a white rectangular input field containing the number "3". At the bottom, there is a label "Test output:" in a dark gray font. Below it is a dark gray rectangular output field containing the number "3".

Рисунок 1 - Результат работы программы

Задача 2: Дано число $1 \leq n \leq 10^7$ необходимо найти последнюю цифру n -го числа Фибоначчи.

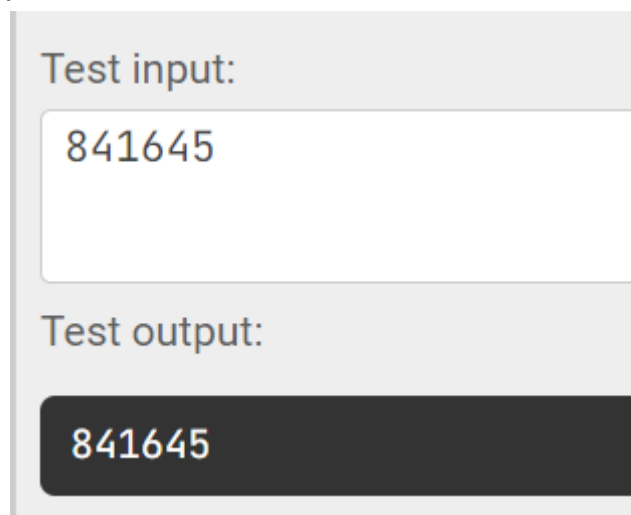
Решение

```
#include <cassert>
#include <iostream>

class Fibonacci {
public:
    static int get_last_digit(int n) {
        assert(n >= 1);
        // put your code here
        return n;
    }
};

int main(void) {
    int n;
    std::cin >> n;
    std::cout << Fibonacci::get_last_digit(n) << std::endl;
    return 0;
}
```

Листинг программы - Подсчет последнего числа числа фибоначчи



The image shows a graphical user interface for a program. It has a light gray background. At the top, there is a label "Test input:" in a dark gray font. Below it is a white rectangular input field with rounded corners, containing the number "841645". Below the input field is another label "Test output:" in a dark gray font. Below this label is a dark gray rectangular output field with rounded corners, containing the number "841645" in a light gray font.

Рисунок 2 - Результат работы программы

Задача 3: По данным двум числам $1 \leq a, b \leq 2 \cdot 10^2$ найдите их наибольший общий делитель.

Решение

```
public class MainClass
{
```

```

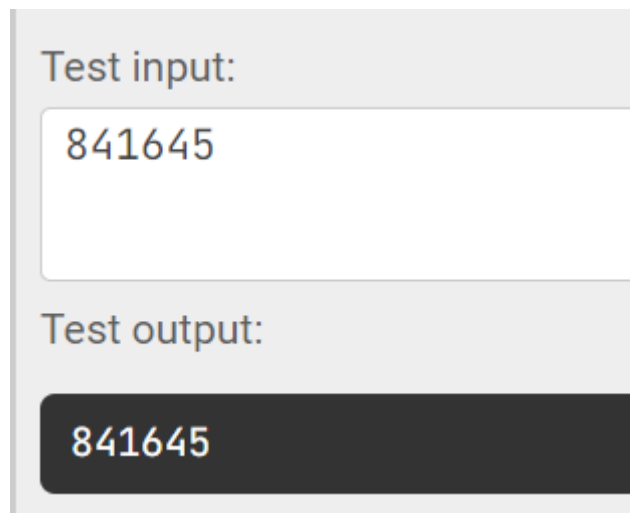
static double EvklidAlgoritm(int a, int b)
{
    if (a == 0)
        return b;
    else if (b == 0)
        return a;
    else if (a >= b)
        return EvklidAlgoritm(a % b, b);
    else
        return EvklidAlgoritm(a, b % a);
}

public static void Main()
{
    string stroka = Console.ReadLine();
    string[] chislo = stroka.Split(' ');

    Console.WriteLine(EvklidAlgoritm(Convert.ToInt32(chisla[0]), Convert.ToInt32(
(chisla[1]))));
}
}

```

Листинг программы - Подсчет наибольшего общего делителя



The image shows a graphical user interface for a program. It has a light gray background. At the top, there is a label "Test input:" in a dark gray font. Below it is a white text input field containing the number "841645". At the bottom, there is a label "Test output:" in a dark gray font. Below it is a dark gray rectangular box containing the number "841645" in a white font.

Рисунок 3 - Результат работы программы

Модуль № 4

“Жадные алгоритмы”

Задача 1: По данным n отрезкам необходимо найти множество точек минимального размера, для которого каждый из отрезков содержит хотя бы одну из точек. В первой строке дано число $1 \leq n \leq 100$ отрезков. Каждая из последующих n строк содержит по два числа, задающих начало и конец отрезка. Выведите оптимальное число m точек и сами m точек. Если таких множеств точек несколько, выведите любое из них.

Решение

```
static class Program
{
    static List<int> Dots(List<Tuple<int, int>> segments)
    {
        List<Tuple<int, int>> res_segments = new List<Tuple<int, int>>();
        while (segments.Count > 0)
        {
            if (segments.Count < 2)
            {
                res_segments.Add(segments.Last());
                segments.RemoveAt(segments.Count - 1);
            }
            else
            {
                Tuple<int, int> a = segments[0], b = segments[1];
                if (b.Item1 <= a.Item2)
                {
                    var left = b.Item1;
                    var right = b.Item2 <= a.Item2 ? b.Item2 : a.Item2;
                    Tuple<int, int> overlapping = new Tuple<int, int>(left,
right);

                    segments = segments.GetRange(2, segments.Count - 2);
                    segments.Insert(0, overlapping);

                }
                else
                {
                    res_segments.Add(segments[0]);
                    segments = segments.GetRange(1, segments.Count - 1);
                }
            }
        }

        List<int> result = new List<int>();
        foreach (var x in res_segments)
        {
            result.Add(x.Item2);
        }
    }
}
```

```

    }

    return result;
}

static void Main()
{
    List<Tuple<int, int>> segments = new List<Tuple<int, int>>();
    int n = int.Parse(Console.ReadLine());

    for (int i = 0; i < n; i++)
    {
        string[] segment = Console.ReadLine().Split(' ');
        segments.Add(new Tuple<int, int>(int.Parse(segment[0]),
int.Parse(segment[1])));
    }

    segments = segments.OrderBy(x => x.Item1).ThenBy(x =>
x.Item2).ToList();
    var result = Dots(segments);

    Console.WriteLine(result.Count);
    result.ForEach(x =>
    {
        Console.Write(x + " ");
    });
}
}

```

Листинг программы - Оптимальное количество точек

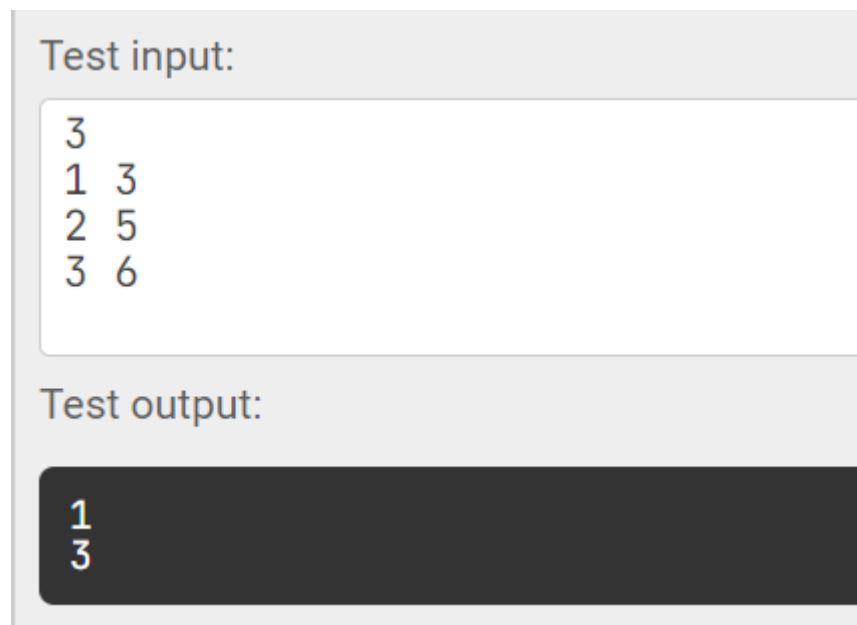


Рисунок 4 - Результат работы программы

Задача 2: Первая строка содержит количество предметов и вместимость рюкзака. Каждая из следующих n строк задает стоимость и объём предмета. Выведите максимальную стоимость частей предметов (от каждого предмета можно отделить любую часть, стоимость и объём при этом пропорционально уменьшатся), помещающихся в данный рюкзак, с точностью не менее трёх знаков после запятой.

Решение

```
using System.Collections.Generic;
using System;
using System.Linq;

namespace ContinuousBackpack
{
    static class Program
    {
        struct Item
        {
            public double Value;
            public double Volume;
            public double ValuePerVolume;

            public Item(string[] input)
            {
                Value = double.Parse(input[0]);
                Volume = double.Parse(input[1]);
                ValuePerVolume = Value / Volume;
            }
        }

        class Backpack
        {
            public double Capacity;
            private double _value = 0;

            public Backpack(string capacity)
            {
                Capacity = double.Parse(capacity);
            }

            public double CalculateValue(List<Item> items)
            {
                while (items.Count > 0)
                {
                    if (Capacity > 0)
                    {
                        if (Capacity >= items[0].Volume)
                        {

```

```

        _value += items[0].Value;
        Capacity -= items[0].Volume;
    }
    else
    {
        _value += Capacity * items[0].ValuePerVolume;
        break;
    }
}
items = items.GetRange(1, items.Count - 1);
}

return _value;
}
}

static void Main()
{
    string[] backpackInput = Console.ReadLine().Split(' ');
    List<Item> items = new List<Item>();

    Backpack backpack = new Backpack(backpackInput[1]);
    for (int i = 0; i < int.Parse(backpackInput[0]); i++)
    {
        items.Add(new Item(Console.ReadLine().Split(' ')));
    }

    items = items.OrderByDescending(x => x.ValuePerVolume).ToList();
    Console.WriteLine($"{backpack.CalculateValue(items):F3}");
}
}
}

```

Листинг программы- Решение задачи

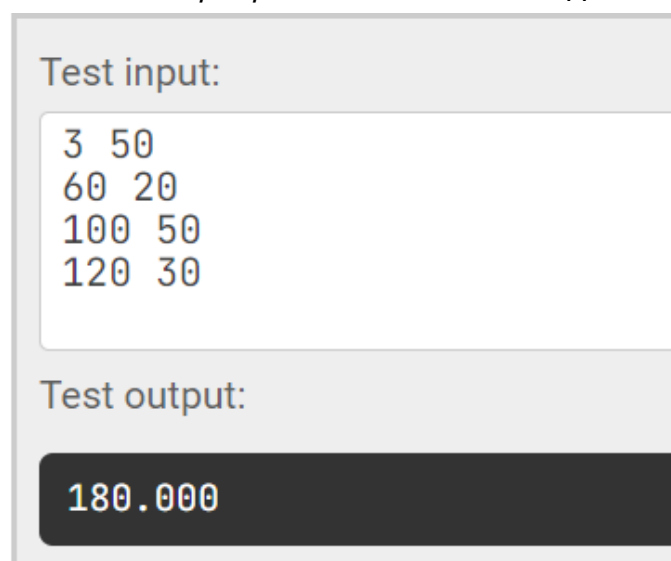


Рисунок 5 - Результат работы программы

Задача 3: По данному числу $1 \leq n \leq 10$ найдите максимальное число k , для которого n можно представить как сумму k различных натуральных слагаемых. Выведите в первой строке число k , во второй – k слагаемых.

Решение

```
using System.Collections.Generic;
using System;
using System.Linq;

static class Program
{
    static void Main()
    {
        int n = int.Parse(Console.ReadLine());
        List<int> k = new List<int>();

        int i = 1;
        int sum = 0;
        while (sum < n)
        {
            k.Add(i);
            sum += i;
            if (sum > n)
            {
                sum -= k.Last();
                k.RemoveAt(k.Count - 1);
                while (sum != n)
                {
                    sum -= k[k.Count - 1];
                    k[k.Count - 1] = k[k.Count - 1] + 1;
                    sum += k[k.Count - 1];
                }
            }

            i++;
        }

        Console.WriteLine(k.Count);
        k.ForEach(x =>
        {
            Console.Write($"{x} ");
        });
    }
}
```

Листинг программы - Решение задачи

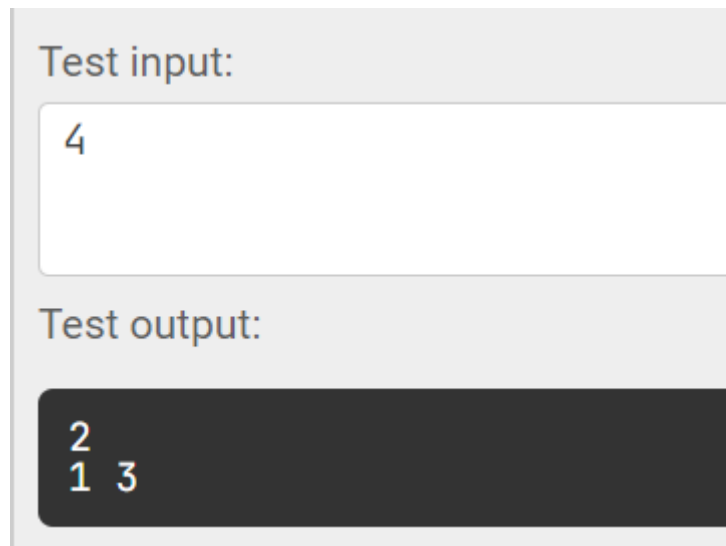


Рисунок 6 - Результат работы программы

Задача 4: По данной непустой строке ss длины не более 10^4 , состоящей из строчных букв латинского алфавита, постройте оптимальный беспрефиксный код. В первой строке выведите количество различных букв kk , встречающихся в строке, и размер получившейся закодированной строки. В следующих kk строках запишите коды букв в формате "letter: code". В последней строке выведите закодированную строку.

Решение

```
using System.Net;
using System.Security.Cryptography.X509Certificates;
using System.Linq;
using System.Collections.Generic;
using System;
static class Program
{
    static string Result = string.Empty;
    private class Node
    {
        public char? Value;

        public Node? Left, Right;

        public char VAL;
        public string Code = string.Empty;

        public Node(char value)
        {
            Value = value;
            VAL = value;
        }
    }
}
```

```

    }

    public void AddNeighbour(Node left, Node right)
    {
        Value = null;
        Left = left;
        Right = right;
    }

    public void AddToCode(string dir)
    {
        Code = dir + Code;
        Left?.AddToCode(dir);
        Right?.AddToCode(dir);
    }
}

private static void Main()
{
    string n = Console.ReadLine();

    Dictionary<char, int> letters = new();

    for (int i = 0; i < n.Length; i++)
    {
        // Ищем уникальные символы
        if (!letters.ContainsKey(n[i]))
            // Добавляем в словарь, и считаем количество символов для
приоритета
            letters.Add(n[i], n.Count(x => x == n[i]));
    }

    // отдельный словарь для сортированных узлов
    Dictionary<Node, int> sorted = new();
    foreach (var c in letters.OrderBy(k => k.Value))
    {
        sorted.Add(new Node(c.Key), c.Value);
    }

    /*
    foreach (var c in sorted)
    {
        Console.WriteLine(c.Key.Value + " " + c.Value);
    }
    */

    Dictionary<Node, int> saveNodes = new();

    if (sorted.Count == 1)
    {

```

```

    var s = sorted.First();

    s.Key.Code = "0";
    saveNodes.Add(s.Key, s.Value);
}

while (sorted.Count != 1)
{
    // Выделяем минимальные узлы, сохраняем и удаляем их
    var left = MinBy(sorted);
    sorted.Remove(left.Key);
    var right = MinBy(sorted);
    sorted.Remove(right.Key);

    var temp = new Node('t');
    temp.AddNeighbour(left.Key, right.Key);
    sorted.Add(temp, left.Value + right.Value);

    left.Key.AddToCode("1");
    right.Key.AddToCode("0");

    if (left.Key.Value is not null)
    {
        saveNodes.Add(left.Key, left.Value);
    }

    if (right.Key.Value is not null)
    {
        saveNodes.Add(right.Key, right.Value);
    }

    // Снова сортируем по возрастанию
    sorted = SortByValue(sorted);
}

Dictionary<char, string> map = new();
foreach (var sn in saveNodes.OrderByDescending(x => x.Value))
{
    map.Add(sn.Key.VAL, sn.Key.Code);
}

string res = String.Empty;
foreach (var v in n)
{
    res += map[v];
}
Console.WriteLine($"{letters.Count} {res.Length}");
foreach (var sn in saveNodes.OrderByDescending(x => x.Value))
{
    Console.WriteLine($"{sn.Key.Value}: {sn.Key.Code}");
}

```

```

        Console.WriteLine(res);
    }

    private static Dictionary<Node, int> SortByValue(Dictionary<Node, int>
letters)
    {
        Dictionary<Node, int> sorted = new();
        foreach (var c in letters.OrderBy(k => k.Value))
        {
            sorted.Add(c.Key, c.Value);
        }

        return sorted;
    }

    private static KeyValuePair<Node, int> MinBy(Dictionary<Node, int>
letters)
    {
        Dictionary<Node, int> sorted = new();
        foreach (var c in letters.OrderBy(k => k.Value))
        {
            return new KeyValuePair<Node, int>(c.Key, c.Value);
        }

        return new KeyValuePair<Node, int>();
    }
}

```

Листинг программы - Решение задачи

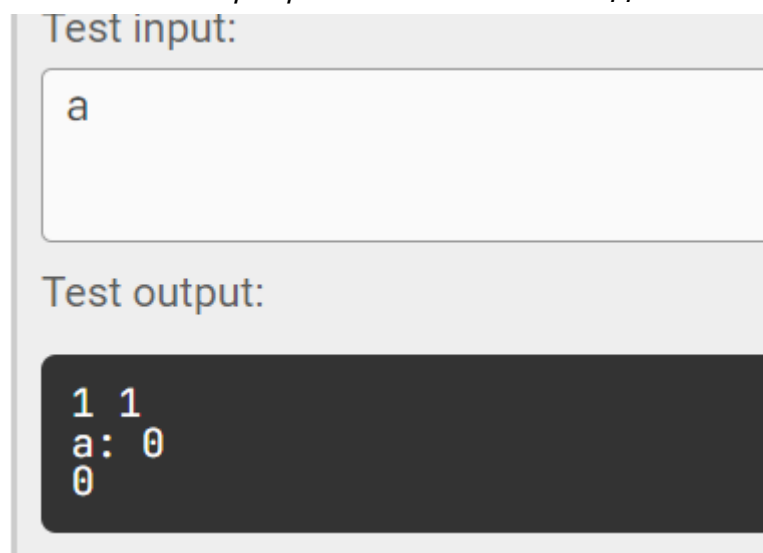


Рисунок 7 - Результат работы программы

Задача 5: Восстановите строку по её коду и беспрефиксному коду символов. В первой строке входного файла заданы два целых числа k и l через пробел — количество различных букв, встречающихся в строке, и размер получившейся закодированной строки, соответственно. В следующих k строках записаны коды букв в формате "letter: code". Ни один код не является префиксом другого. Буквы могут быть перечислены в любом порядке. В качестве букв могут встречаться лишь строчные буквы латинского алфавита; каждая из этих букв встречается в строке хотя бы один раз. Наконец, в последней строке записана закодированная строка. Исходная строка и коды всех букв непусты. Заданный код таков, что закодированная строка имеет минимальный возможный размер.

Решение

```
using System.Collections.Generic;
using System.Linq;

public class Program
{
    public static void Main()
    {
        Dictionary<string, char> map = new Dictionary<string, char>();

        string lengths = Console.ReadLine();
        var l = lengths.Split(" ");
        int uniqueChars = Int32.Parse(l.First());

        for (int i = 0; i < uniqueChars; i++)
        {
            var input = Console.ReadLine().Split(": ");
            map.Add(input.Last(), input.First()[0]);
        }

        string coded = Console.ReadLine();
        string decoded = String.Empty;

        string token = string.Empty;
        foreach (var ch in coded)
        {
            token += ch;
            if (map.ContainsKey(token))
            {
                decoded += map[token];
                token = String.Empty;
            }
        }
    }
}
```

```

        Console.WriteLine(decoded);
    }
}

```

Листинг программы - Решение задачи

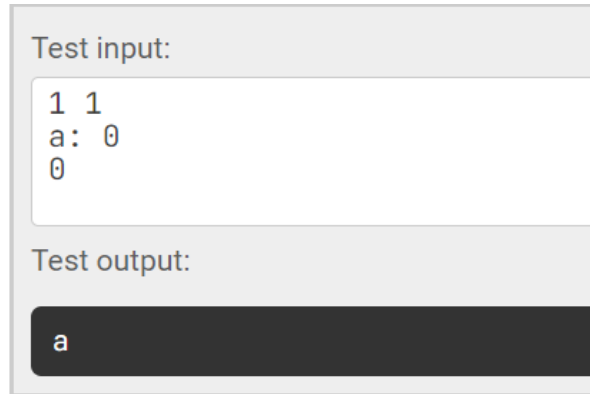


Рисунок 8 - Результат работы программы

Задача 5: Первая строка входа содержит число операций n . Каждая из последующих n строк задают операцию одного из следующих двух типов:

- Insert x — целое число;
- ExtractMax.

Первая операция добавляет число x в очередь с приоритетами, вторая — извлекает максимальное число и выводит его.

Решение

```

using System;
using System.Collections;

public class MainClass
{
    public static void Main()
    {
        var input = Console.ReadLine();
        var commandCount = int.Parse(input);

        PriorityQueue<int> queue = new PriorityQueue<int>();
        for(int i=0; i<commandCount; i++)

```

```

{
    var commands = Console.ReadLine()!.Split(' ');
    switch (commands[0])
    {
        case "Insert":
        {
            var value = int.Parse(commands[1]);
            queue.Add(value, value); break;
        }
        case "ExtractMax": Console.WriteLine(queue.Poll()); break;
    }
}

}

class PriorityQueue<T>
{
    struct Item<T>
    {
        public T value;
        public int priority;

        public Item(T value, int priority)
        {
            this.value = value;
            this.priority = priority;
        }

        public bool Compare(Item<T> item)
        {
            return priority > item.priority;
        }
    }

    static ArrayList items = new ArrayList();

    private void BinAdd(Item<T> item)
    {
        int i = -1;
        int j = items.Count;
        int avr;
        while(i + 1 < j)
        {
            avr = (i + j) >> 1;
            if(item.Compare((Item<T>)items[avr])) {
                j = avr;
            }
            else i = avr;
        }
        items.Insert(++i, item);
    }

    // Return the number of items in the queue.
    public int Count

```



```

{
    get
    {
        return items.Count;
    }
}

// Add an item to the queue.
public void Add(T new_value, int new_priority)
{
    BinAdd(new Item<T>(new_value, new_priority));
}

// Remove the item with the largest priority from the queue.
public T Poll()
{
    Item<T> top_item = (Item<T>)items[0];
    items.Remove(top_item);
    return top_item.value;
}
}
}

```

Листинг программы - Решение задачи

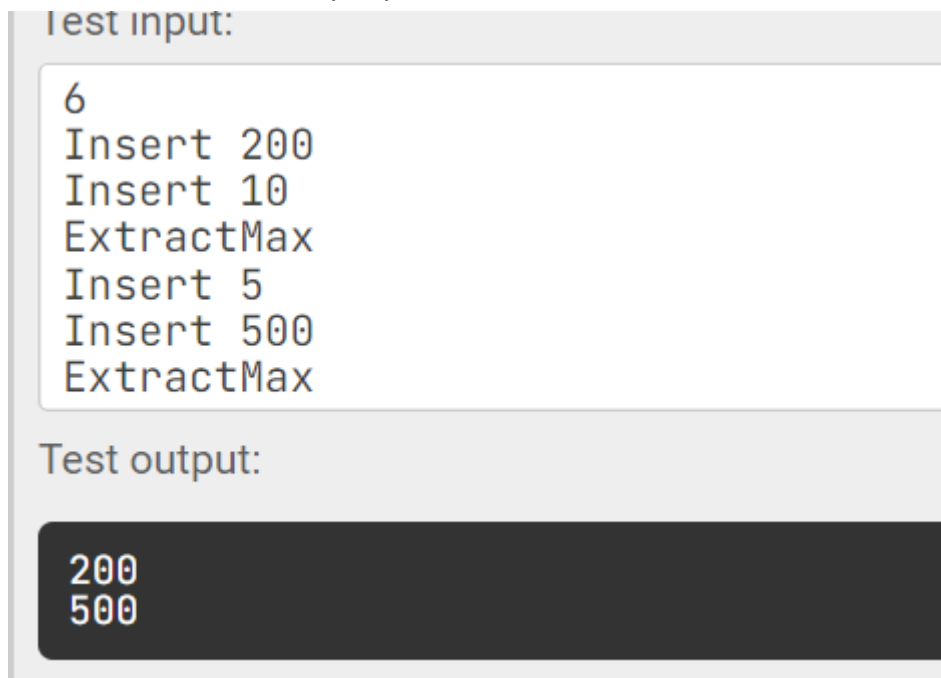


Рисунок 9 - Результат работы программы

Модуль № 6

“Разделяй и властвуй”

Задача 1: В первой строке даны целое число $1 \leq n \leq 10$ и массив $A[1 \dots n]$ из n различных натуральных чисел, в порядке возрастания, во второй – целое число, не превышающих 10^9 . Для каждого i от 1 до k необходимо вывести индекс $1 \leq j \leq n$, для которого $A[j] = b_i A[j] = b$, или -1, если такого j вывести “нет”.

Решение

```
using System;
using System.Collections.Generic;

public class MainClass
{
    private static Dictionary<int,int> FillList() {
        string[] input = Console.ReadLine().Split(' ');
        Dictionary<int,int> list = new Dictionary<int,int>();
        for(int i = 1; i <= int.Parse(input[0]); i++) {
            list.Add(int.Parse(input[i]), i);
        }

        return list;
    }

    private static Dictionary<int,int> FillListUn() {
        string[] input = Console.ReadLine().Split(' ');
        Dictionary<int,int> list = new Dictionary<int,int>();
        for(int i = 1; i <= int.Parse(input[0]); i++) {
            list.Add(i, int.Parse(input[i]));
        }

        return list;
    }

    public static void Main()
    {
        Dictionary<int,int> x = FillList();
        Dictionary<int,int> y = FillListUn();

        foreach(KeyValuePair<int,int> i in y) {
            if (x.ContainsKey(i.Value)) {
                Console.Write($"{x[i.Value]} ");
            }
            else
            {
                Console.Write("-1 ");
            }
        }
    }
}
```

Листинг программы- Двоичный поиск

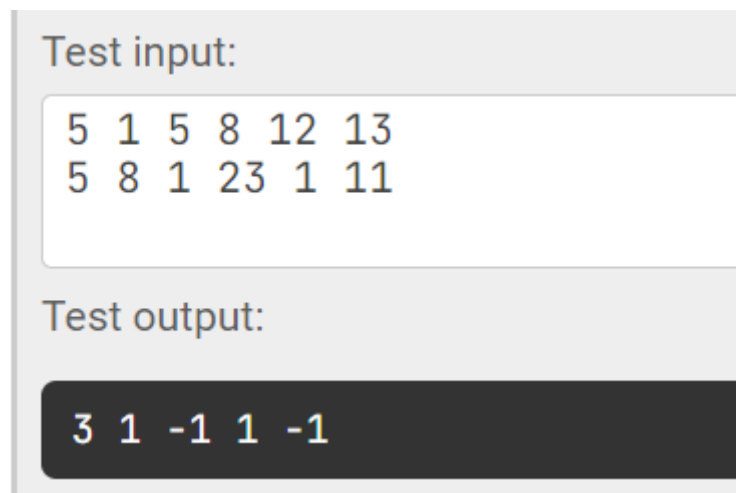


Рисунок 10 - Результат работы программы

Задача 2: Первая строка содержит число $1 \leq n \leq 10^5$, вторая — массив $A[1 \dots n]$, содержащий натуральные числа, не превосходящие 10^9 .

Необходимо посчитать число пар индексов $1 \leq i < j \leq n$, для которых $A[i] > A[j]$. (Такая пара элементов называется инверсией массива. Количество инверсий в массиве является в некотором смысле его мерой неупорядоченности: например, в упорядоченном по неубыванию массиве инверсий нет вообще, а в массиве, упорядоченном по убыванию, инверсию образуют каждые два элемента.)

Решение

```
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
#include <queue>

using namespace std;

template <typename Type>
vector<Type> fill_cont() {
    istream_iterator<Type> it{cin};
    istream_iterator<Type> eos;
    vector<Type> v{it, eos};

    return v;
}

template <typename It>
vector<typename It::value_type> merge(const It begin, const It mid, const
It end, long long& inv) {
    vector<typename It::value_type> v;
    It first_arr{begin};
```

```

It second_arr{mid};
const It first_arr_end{mid};
const It second_arr_end{end};

while (first_arr != first_arr_end && second_arr != second_arr_end) {
    if (*first_arr <= *second_arr) {
        v.push_back(*first_arr++);
    }
    else {
        v.push_back(*second_arr++);
        inv += distance(first_arr, first_arr_end);
    }
}

v.insert(v.end(), first_arr, first_arr_end);
v.insert(v.end(), second_arr, second_arr_end);

return v;
}

template <typename It>
auto merge_sort(It begin, It end) -> long long {
    auto size = distance(begin, end);
    long long inv = 0;

    if (size > 1) {
        auto mid = next(begin, size / 2);

        inv += merge_sort(begin, mid);
        inv += merge_sort(mid, end);

        auto &&v = merge(begin, mid, end, inv);
        move(v.begin(), v.end(), begin);
    }

    return inv;
}

template <typename Index, typename Cont, typename Inv_type>
pair<Index, Cont> merge_it(const pair<Index, Cont>& left, const pair<Index,
Cont>& right, Inv_type& inv) {
    Cont v;
    typename Cont::const_iterator f_it{left.second.begin()};
    typename Cont::const_iterator f_end{left.second.end()};
    typename Cont::const_iterator s_it{right.second.begin()};
    typename Cont::const_iterator s_end{right.second.end()};

    while (f_it != f_end && s_it != s_end) {
        if (*f_it <= *s_it) {
            v.push_back(*f_it++);
        }
        else {
            v.push_back(*s_it++);

```

```

        inv += distance(f_it, f_end);
    }
}

v.insert(v.end(), f_it, f_end);
v.insert(v.end(), s_it, s_end);

return make_pair(min(left.first, right.first), v);
}

template <typename It>
auto iterative_merge_sort(It begin, It end) -> long long {
    using queue_elem = pair<int, vector<typename It::value_type>>;
    queue<typename It::value_type, deque<queue_elem>> q;
    long long inv = 0;
    It init_it = begin;

    for (auto i = 0; init_it != end; ++init_it, ++i) {
        q.push(make_pair(i, vector<typename It::value_type>{*init_it}));
    }

    while(q.size() > 1) {
        auto first = q.front();
        q.pop();
        auto second = q.front();

        // if a given element should go after others
        if (first.first > second.first) {
            q.push(first);
        }
        else {
            q.pop();
            q.push(merge_it(first, second, inv));
        }
    }

    copy(q.front().second.begin(), q.front().second.end(), begin);

    return inv;
}

int main() {
    int n;
    pair<int, vector<char>> p;

    cin >> n;
    auto v = fill_cont<int>();
    auto res = merge_sort(v.begin(), v.end());
    // auto res = iterative_merge_sort(v.begin(), v.end());

    cout << res << endl;

    return 0;
}

```

```
}
```

Листинг программы- Подсчет инверсий

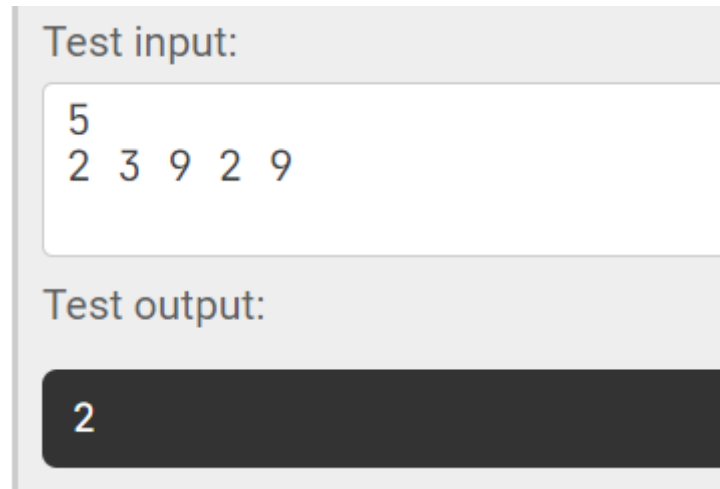


Рисунок 11 - Результат работы программы

Задача 3: В первой строке задано два целых числа $1 \leq n \leq 50000$ и $1 \leq m \leq 50000$ — количество отрезков и точек на прямой, соответственно. Следующие n строк содержат по два целых числа a_i — координаты концов отрезков. Последняя строка содержит m целых чисел — координаты точек. Все координаты не превышают 10^8 по модулю. Точка считается принадлежащей отрезку, если она находится внутри него или на границе. Для каждой точки в порядке появления во вводе выведите, скольким отрезкам она принадлежит.

Решение

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <sstream>

int main()
{
    // Количество отрезков и точек
    unsigned int num_lines = 0, num_points = 0;
    // массив отрезков
    std::vector<std::pair<int, int>> lines;
    // массив точек
    std::vector<std::pair<int, int>> points;
    std::cin >> num_lines >> num_points;
    while (num_lines-- >= 1) {
```

```

    int a = 0, b = 0;
    std::cin >> a >> b;
    lines.push_back(std::make_pair(a, b));
}
int counter = 0;
while (counter < num_points) {
    int pt = 0;
    std::cin >> pt;
    points.push_back(std::make_pair(counter, pt));
    ++counter;
}
// Сортируем массив точек
std::stable_sort(points.begin(), points.end(), [](const std::pair<int,
int> &e1, const std::pair<int, int> &e2) {return e1.second < e2.second;});
// Сортируем массив отрезков по левому краю
std::stable_sort(lines.begin(), lines.end(), [](const std::pair<int, int>
&e1, const std::pair<int, int> &e2) {return e1.first < e2.first;});
// Создаем новый массив из отрезков и упорядочиваем его по правому краю
std::vector<std::pair<int, int>> part;
part.insert(part.begin(), lines.begin(), lines.end());
std::stable_sort(part.begin(), part.end(), [](const std::pair<int, int>
&e1, const std::pair<int, int> &e2) {return e1.second < e2.second;});
// Массив числа отрезков содержащих каждую точку
std::vector<std::pair<int, int>> numbers;
num_points = points.size();
// Выбираем первую точку
auto i_pt = points.begin();
// Оптимизируем для одинаковых точек ???
int prev_pt = (*i_pt).second;
int prev_num = 0;
bool is_prev = false;
auto prev_iter_left = lines.begin();
auto prev_iter_right = part.begin();
while (num_points > numbers.size()) { // Пока пройдены не все точки
    if (is_prev && (*i_pt).second == prev_pt) {
        numbers.push_back(std::make_pair((*i_pt).first, prev_num));
        i_pt++;
        continue;
    }
    // Берем первую точку и находим позицию первого отрезка левый край
    // которого более точки
    auto pos_left_line = std::find_first_of(prev_iter_left, lines.end(),
i_pt, i_pt + 1, [](const std::pair<int, int> &e1, const std::pair<int, int>
&e2) {return e1.first > e2.second;});
    if (pos_left_line == lines.begin()) { // Начала всех отрезков правее
    точки - пересечений нет
        numbers.push_back(std::make_pair((*i_pt).first, 0));
        is_prev = true;
        prev_pt = (*i_pt).second;
        prev_num = 0;
        i_pt++;
        continue;
    }
}

```

```

    prev_iter_left = lines.begin() == pos_left_line ? pos_left_line :
pos_left_line - 1;
    // Находим позицию первого отрезка правый край которого более либо
    // равен точки
    auto pos_right_line = std::find_first_of(prev_iter_right, part.end(),
i_pt, i_pt + 1, [](const std::pair<int, int> &e1, const std::pair<int, int>
&e2) {return e1.second >= e2.second;});
    prev_iter_right = part.begin() == pos_right_line ? pos_right_line :
pos_right_line - 1;
    // Сохраняем найденное число отрезков
    numbers.push_back(std::make_pair((*i_pt).first, (pos_left_line -
lines.begin()) - (pos_right_line - part.begin())));
    is_prev = true;
    prev_pt = (*i_pt).second;
    prev_num = (pos_left_line - lines.begin()) - (pos_right_line -
part.begin());
    i_pt++;
}
// Выводим количество найденных (оставшихся отрезков)
std::ostringstream oss;
// Сортируем массив точек
std::stable_sort(numbers.begin(), numbers.end(), [](const std::pair<int,
int> &e1, const std::pair<int, int> &e2) {return e1.first < e2.first;});
for (auto elem : numbers) { oss << elem.second << " "; }
std::cout << oss.str() << std::endl;

return 0;
}

```

Листинг программы - Точки и отрезки

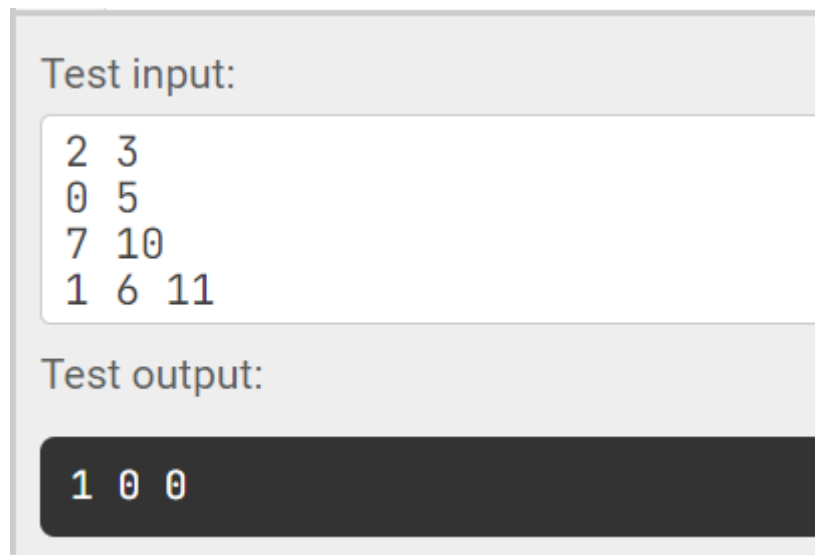


Рисунок 12 - Результат работы программы

Задача 4: Первая строка содержит число $1 \leq n \leq 10^4$, вторая — n натуральных чисел, не превышающих 10. Выведите упорядоченную по не убыванию последовательность этих чисел.

Решение

```
#include <iostream>
#include <sstream>
#include <vector>

int main()
{
    int n = 0;
    std::cin >> n;
    std::vector<unsigned int> b(11);
    while (--n >= 0) {
        unsigned int value = 0;
        std::cin >> value;
        b[value] += 1;
    }
    std::ostringstream oss;
    for (unsigned int i = 0; i < 11; ++i)
        for (unsigned int j = 0; j < b[i]; ++j) oss << i << " ";
    std::cout << oss.str() << std::endl;
    return 0;
}
```

Листинг программы- сортировка подсчётом

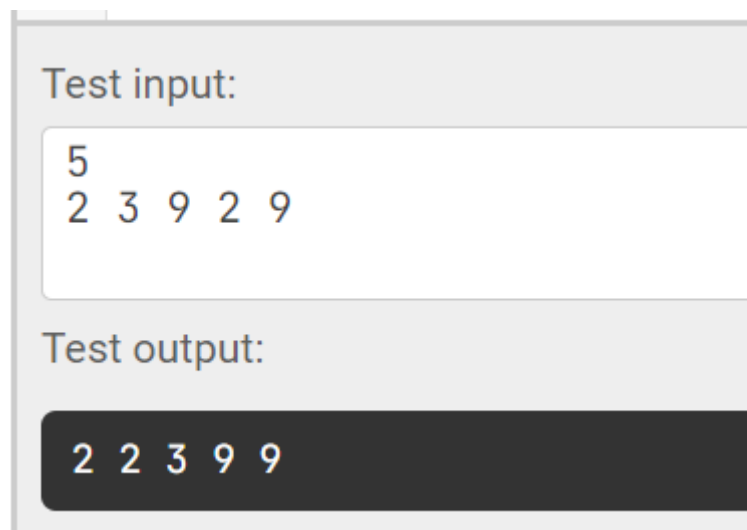


Рисунок 13 - Результат работы программы