Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Иркутский национальный исследовательский технический

университет»

Институт информационных технологий и анализа данных

**О Т Ч Ё Т**

|  |  |  |
| --- | --- | --- |
| о прохождении | | учебной практики |
|  | | (вид практики: учебная/производственная) |
| технологической (проектно-технологической) практики | | |
| (тип практики: технологическая/научно-исследовательская работа/преддипломная и др.) | | |
|  | | |
| в | ИРНИТУ | |
|  | (наименование профильной организации) | |

Обучающегося Султангараева А.И., ИСИб-24-1

(ФИО, группа, подпись)

Руководитель практики от института ИТиАД

Кононенко Р.В., доцент института ИТиАД

(ФИО, должность, подпись)

Ссылка на резюме обучающего на сайте https://hh.ru/

Руководитель образовательной программы

Кононенко Р.В., доцент института ИТиАД

(ФИО, должность, подпись)

Оценка по практике

(ФИО, подпись, дата)

Содержание отчета на стр.

Иркутск 2025

**Индивидуальное задание на прохождение**

**учебной практики: технологической (проектно-технологической) практики**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Султангараевой Альбины Ильсафовны | | | | |
|  | (ФИО обучающегося полностью) | | | | |
| обучающегося | | 1 | курса | группы | ИСИб-24-1 |

по направлению подготовки Информационные системы и технологии

профиль Интеллектуальные системы обработки информации и управления

Место прохождения практики: ИРНИТУ

Сроки прохождения практики с «16» июня 2025 г. по «29» июня 2025 г.

Цели и задачи прохождения практики: решить задачи и проверить на разных данных, что работает.

Содержание практики, вопросы, подлежащие изучению: решение задач и их проверка на разных данных.

Планируемые результаты практики: научиться решать новые задачи, закрепить знания, полученные в течение семестра.

Руководитель практики от

института ИТиАД

\_\_\_\_\_\_\_\_\_\_\_/Кононенко Р.В./

(подпись)

**Согласовано:**

Руководитель ООП

\_\_\_\_\_\_\_\_\_\_\_/Кононенко Р.В./

(подпись)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2025 г.

С настоящим индивидуальным заданием и с программой практики ознакомлен(а), задание принято к исполнению

\_\_\_\_\_\_\_\_\_\_ «16» июня 2025 г.

(подпись)

Ход решения задачи 1

Условие:

Незнайка в своей экспедиции на Луну оказался на вершине лунной горы. Спуск вниз опасен, поэтому он взял с собой карту склона горы, где числами обозначено, сколько минут требуется на этот участок маршрута. Спуск происходит сверху вниз на один из соседних участков. Пример наиболее короткого маршрута выделен красным цветом, сумма чисел = 10.



Рисунок 1 – Пример пирамиды

Напишите программу, рассчитывающую минимальное время спуска (сумму чисел в пути с вершины до основания).

**Формат входных данных**

В первой строке дано целое число N - высота пирамиды, далее следуют N строк из чисел, разделённых пробелом (в каждой строке на 1 число больше, чем в предыдущей)

**Формат выходных данных**

Сумма чисел в пути с вершины до основания (одно число)

Последовательность участков маршрута (числа, разделённые пробелом)



Рисунок 2 – Пример теста для задачи 1

Решение задачи 1 на языке C++:

#include <iostream>

#include <vector>

int main() {

int n;

std::cout << "Введите высоту пирамиды: ";

std::cin>>n;

std::vector<std::vector<int>> pyr(n);

for (int i = 0; i < n; i++) {

for (int j = 0; j <= i; j++) {

pyr[i].push\_back(std::rand() % 1000);

}

}

std::cout << "\nПирамида: " << '\n';

for (int i = 0; i < n; i++) {

for (int j = 0; j <= i; j++) {

std::cout << pyr[i][j] << " ";

}

std::cout << '\n';

}

std::vector<std::vector<int>> sum = pyr;

for (int i = n - 2; i >= 0; i--) {

for (int j = 0; j < i; j++ ) {

sum[i][j] += std::min(sum[i+1][j], sum[i+1][j+1]);

}

}

std::cout << "Минимальная сумма: " << sum[0][0] << '\n';

std::cout << pyr[0][0] << " ";

int j = 0;

for (int i = 1; i < n; i++) {

if (sum[i][j] < sum[i][j+1]) {

std::cout << pyr[i][j] << " ";

} else {

std::cout << pyr[i][j+1] << " ";

j++;

}

}

return 0;

}

Тесты:

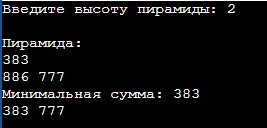


Рисунок 3 – Тест 1

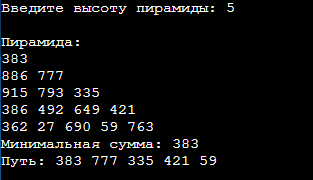


Рисунок 4 – Тест 2

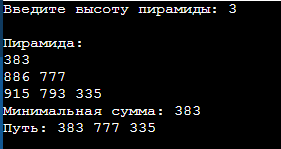


Рисунок 5 – Тест 3

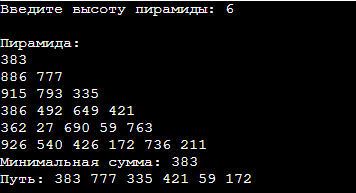


Рисунок 6 – Тест 4

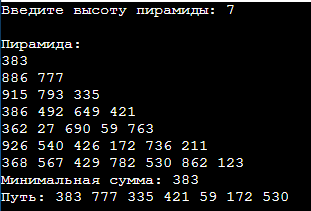


Рисунок 7 – Тест 5

Ход решения задачи 2

Условие:

После метеоритной атаки компьютерная сеть для управления лунными заводами разбилась на части, нужно объединить её в единое целое. Каждый фрагмент сети представлен в виде ненаправленного графа.

Вам известно общее число вершин графа (узлы сети, не более 1000) и набор рёбер (сохранившиеся линии связи, не более 1000).

Определите, какое минимальное число линий связи нужно дополнительно построить, чтобы сеть стала единой.



Рисунок 8 – Пример графа



Рисунок 9 – Рёбра графа

**Формат входных данных**

В первой строке дано целое число N - количество узлов сети и M - число линий связи. Далее следуют M строк из чисел, разделённых пробелом (узлы, которые связывает данная линия)

**Формат выходных данных**

Число необходимых линий связи (одно число)



Рисунок 10 – Пример теста для задачи 2

Решение задачи 2 на C++:

#include <iostream>

#include <vector>

#include <queue>

int main() {

std::cout << "Введите количество узлов сети и количество соединений: ";

int N, M;

std::cin >> N >> M;

std::vector<std::vector<int>> adj(N + 1);

std::cout << "\nВведите " << M << " пар соединений (узлы, связанные линией):\n";

for (int i = 0; i < M; ++i) {

std::cout << "Соединение " << (i + 1) << ": ";

int u, v;

std::cin >> u >> v;

adj[u].push\_back(v);

adj[v].push\_back(u);

}

std::vector<bool> visited(N + 1, false);

int components = 0;

for (int i = 1; i <= N; ++i) {

if (!visited[i]) {

components++;

std::queue<int> q;

q.push(i);

visited[i] = true;

while (!q.empty()) {

int u = q.front();

q.pop();

for (int v : adj[u]) {

if (!visited[v]) {

visited[v] = true;

q.push(v);

}

}

}

}

}

std::cout << "\nМинимальное количество линий для соединения сети: ";

std::cout << (components - 1) << "\n";

return 0;

}

Тесты:

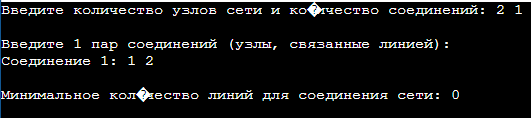


Рисунок 10 – Тест 1 для задачи 2

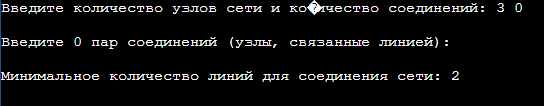


Рисунок 11 – Тест 2 для задачи 2

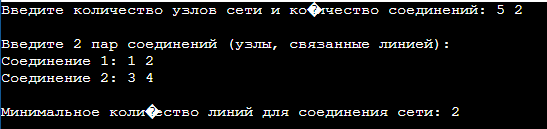


Рисунок 12 – Тест 3 для задачи 2

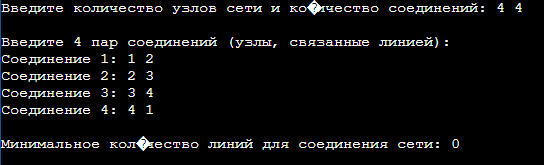


Рисунок 13 – Тест 4 для задачи 2

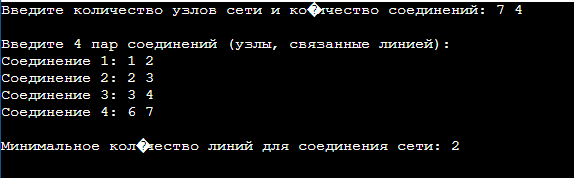


Рисунок 14 – Тест 5 для задачи 2

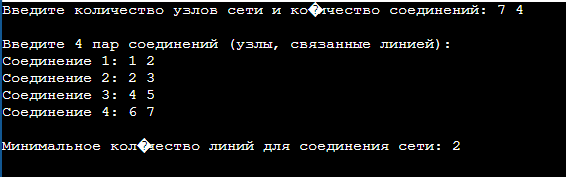


Рисунок 15 – Тест 6 для задачи 2

Ход решения задачи 3

Условие:

В Иркутске раз в году наступает зима. Не смотря на то что событие это довольно регулярное, оно всегда внезапно. Снег буквально заваливает все улицы, не давая проехать на чём-то меньше трактора. В этом году терпение лопнуло и специальным указом был создан кризисный центр по борьбе с сугробами. Центру были переданы спутники, лазеры, метеорологические зонды и несколько десятков лопат.

Вам поручено возглавить отдел разведки снежной ситуации и быть способным чрезвычайно быстро отвечать на запросы центра. Сам город состоит из нескольких, расположенных подряд, улиц, каждая из которых абсолютна похожа на любую другую.

* Информация о снеге передается вам в виде тройки чисел – 1 в качестве идентификатора события, уникального индекса улицы и количество миллиметров выпавшего снега.
* Запросы в свою очередь так же имеют вид тройки чисел – 2 в качестве идентификатора события, индекс улицы с которой нужно суммировать количество выпавшего снега и индекс улицы по которую нужно суммировать, крайние улицы должны быть включены.

**Формат входных данных**

Первая строка входных данных содержит два целых числа – n (1 или больше) и k (0 или больше) это количество чисел в массиве и количество запросов соответственно.

Следующие k строк содержат:

* либо 1 i x – Учетная информация о количестве, выпавшего на улице i (больше 0) x миллиметров снега.
* либо 2 u r – Запрос на подсчет количества снега на улицах от u до r (u и r больше 0 и могут быть равны друг другу)

**Формат выходных данных**

На каждый запрос второго типа надо вывести единственное число – суммарное выпавшего на них снега с момента начала наблюдения.



Рисунок 15 – Пример теста для задачи 3

Решение задачи 3 на языке C++:

#include <iostream> //задача 3

#include <vector>

using namespace std;

int main() {

cout << "Введите количество улиц и количество запросов: ";

int n, k;

cin >> n >> k;

vector<int> streets(n + 1, 0);

for (int i = 0; i < k; ++i) {

cout << "\nЗапрос " << (i + 1) << " из " << k << ": ";

int type, a, b;

cin >> type >> a >> b;

if (type == 1) {

streets[a] += b;

cout << "Добавлено " << b << " мм снега на улицу " << a << '\n';

} else if (type == 2) {

int sum = 0;

for (int j = a; j <= b; ++j) {

sum += streets[j];

}

cout << "Сумма снега с улицы " << a << " по " << b << " = " << sum << '\n';

} else {

cout << "Напишите 1 или 2\n";

--i;

}

}

return 0;

}

Тесты:

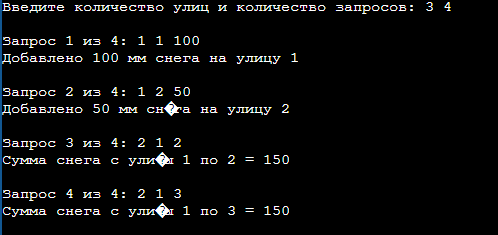


Рисунок 16 – Тест 1 для задачи 3

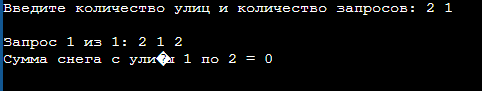


Рисунок 17 – Тест 2 для задачи 3

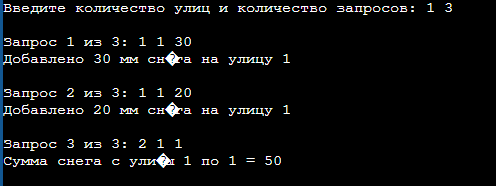


Рисунок 18 – Тест 3 для задачи 3

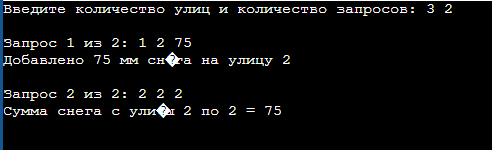


Рисунок 19 – Тест 4 для задачи 3

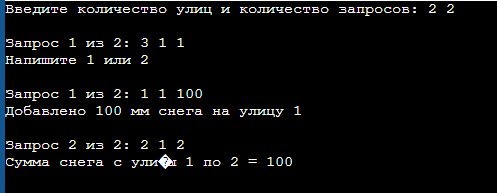


Рисунок 20 – Тест 5 для задачи 3

Ход решения задачи 4

Условие:

Перестановка P длины n — это упорядоченный набор, содержащий числа от 1 до n, каждое из которых входит в него ровно один раз. Например, перестановкой длины 13 является набор (5 11 13 12 6 1 8 4 10 9 7 2 3). Само название говорит о том, для чего предназначен этот объект. Например, можно при помощи перестановки букв зашифровать слово. Для примера возьмем приведенную выше перестановку и слово transposition, которое состоит тоже из 13 букв. Далее, следуя перестановке, на первую позицию поставим пятую букву слова, на вторую − одиннадцатую букву и так далее. В итоге получим sinoptsntiora. К этому слову снова применим эту же перестановку и получим poartsnoitsin. Повторив эти стадии шифрования k раз, получим зашифрованное сообщение.



Рисунок 21 – Пример перестановки

Вам дано зашифрованное таким образом слово, шифрующая перестановка P и число k. Необходимо восстановить слово.

**Формат входных данных**

Первая строка входных данных содержит 2 числа – n и k (1 или больше, могут быть равны). Следующая строка содержит перестановку длиной n, числа разделяются пробелом. Третья строка содержит зашифрованное слово длиной n.

**Формат выходных данных**

Вывести одну строку − исходное слово.



Рисунок 22 – Пример теста для задачи 4

Решение задачи 4 на языке C++:

#include <iostream>

#include <vector>

#include <string>

using namespace std;

vector<int> Permutation(const vector<int>& permutation, const vector<int>& vec) {

vector<int> result(vec.size());

for (int i = 0; i < vec.size(); ++i) {

result[i] = vec[permutation[i] - 1];

}

return result;

}

vector<int> InversePermutation(const vector<int>& permutation) {

vector<int> inverse(permutation.size());

for (int i = 0; i < permutation.size(); ++i) {

inverse[permutation[i] - 1] = i + 1;

}

return inverse;

}

vector<int> permutationPower(const vector<int>& permutation, int power) {

vector<int> current = permutation;

vector<int> result(permutation.size());

for (int i = 0; i < result.size(); ++i) {

result[i] = i + 1;

}

while (power > 0) {

if (power % 2 == 1) {

result = Permutation(current, result);

}

current = Permutation(current, current);

power /= 2;

}

return result;

}

int main() {

cout << "Введите длину слова и количество шифрований: ";

int n, k;

cin >> n >> k;

cout << "Введите перестановку: ";

vector<int> permutation(n);

for (int i = 0; i < n; ++i) {

cin >> permutation[i];

}

cout << "Введите зашифрованное слово: ";

string line;

cin >> line;

vector<int> inverse = InversePermutation(permutation);

vector<int> totalInverse = permutationPower(inverse, k);

string original(n, ' ');

for (int i = 0; i < n; ++i) {

original[i] = line[totalInverse[i] - 1];

}

cout << "Исходное слово: " << original << '\n';

return 0;

}

Тесты:

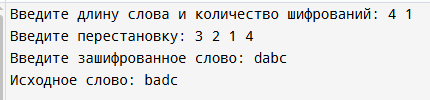


Рисунок 23 – Тест 1 для задачи 4

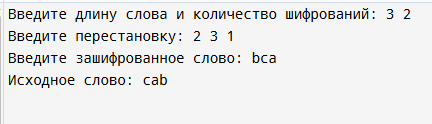


Рисунок 24 – Тест 2 для задачи 4

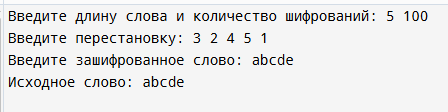


Рисунок 25 – Тест 3 для задачи 4

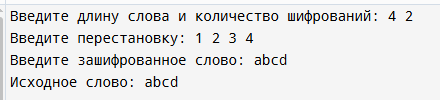


Рисунок 26 – Тест 4 для задачи 4

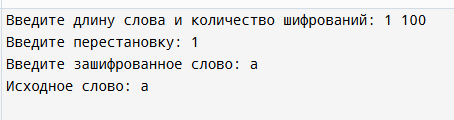


Рисунок 27 – Тест 5 для задачи 4

Ход решения задачи 5

Условие:

Дана матрица, состоящая из 1 и 0. Значениями 1 в матрице нарисована некоторая фигура. Необходимо определить координаты верхнего левого и нижнего правого углов параллельного осям ограничивающего прямоугольника, т.е. такого прямоугольника, минимального размера, в который фигура помещается полностью и при этом ни одна точка исходной фигуры не попадает на стороны прямоугольника.

**Формат входных данных**

В первой строке через пробел заданы высота h и ширина w матрицы (длина и ширина 10 или больше, но не больше 50, могут быть равны). В следующих строках заданы значения матрицы по строкам и столбцам. В матрице всегда есть только одна фигура. Фигура отстоит от краев матрицы минимум на один ноль. Начало координат в левом верхнем углу. Координаты растут вниз и вправо.

**Формат выходных данных**

Координаты верхнего левого и правого нижнего угла прямоугольника отделенные пробелами. Координаты задаются номером строки и номером столбца. Нумерация начинается с 0.



Рисунок 28 - Пример теста для задачи 5

Решение задачи 5 на языке C++:

#include <iostream>

#include <vector>

#include <limits>

int main() {

int h, w;

std::cout << "Введите высоту и ширину матрицы: ";

std::cin >> h >> w;

std::vector<std::vector<int>> matrix(h, std::vector<int>(w));

std::cout << "Введите значения матрицы построчно (только 0 и 1):\n";

for (int i = 0; i < h; ++i) {

std::cout << "Строка " << i << ": ";

for (int j = 0; j < w; ++j) {

std::cin >> matrix[i][j];

}

}

int min\_row = std::numeric\_limits<int>::max();

int max\_row = std::numeric\_limits<int>::min();

int min\_col = std::numeric\_limits<int>::max();

int max\_col = std::numeric\_limits<int>::min();

for (int i = 0; i < h; ++i) {

for (int j = 0; j < w; ++j) {

if (matrix[i][j] == 1) {

if (i < min\_row) min\_row = i;

if (i > max\_row) max\_row = i;

if (j < min\_col) min\_col = j;

if (j > max\_col) max\_col = j;

}

}

}

std::cout << "\nКоординаты ограничивающего прямоугольника:\n";

std::cout << min\_row << " " << min\_col << " " << max\_row << " " << max\_col << "\n";

return 0;

}

Тесты:

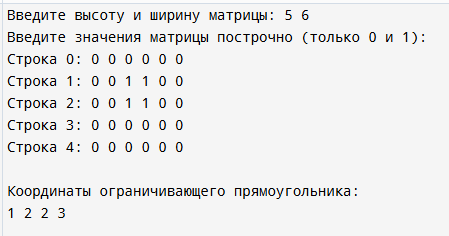


Рисунок 29 – Тест 1 для задачи 5

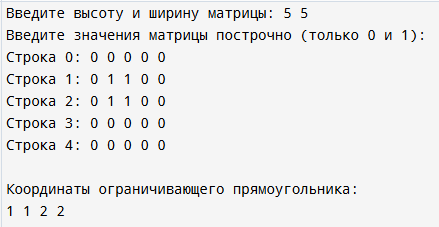


Рисунок 30 – Тест 2 для задачи 5

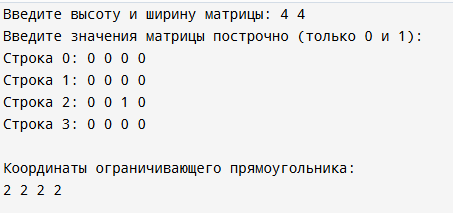


Рисунок 31 – Тест 3 для задачи 5

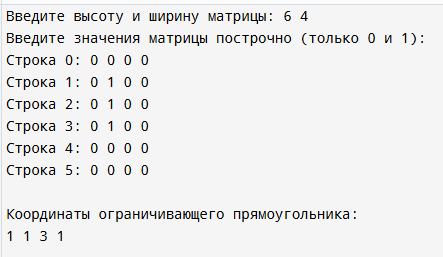


Рисунок 32 – Тест 4 для задачи 5

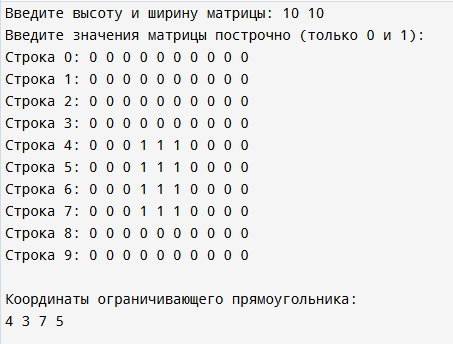


Рисунок 33 – Тест 5 для задачи 5

Ход решения задачи 6

Условие:

В школьном кружке робототехники есть два вида микроконтроллеров (условно тип A и тип B) и два вида модулей управления мотором (условно тип 1 и тип 2). Выяснилось, что контроллер типа B и модуль управления типа 2 несовместимы. Использование микроконтроллеров и модулей управления в других комбинациях возможно. Имеется a микроконтроллеров типа A, b микроконтроллеров типа B, x модулей управления типа 1 и y модулей типа 2. Определите, какое максимальное число работающих пар из микроконтроллера и модуля управления мотором можно составить. Ваша программа должна ответить на n запросов.

**Формат входных данных**

В первой строке пишем число n (не больше 50). Далее в n строках пишем по 4 натуральных числа (a, b, x, y).

**Формат выходных данных**

Выводим n чисел через пробел, каждое число – максимальное число работающих пар из микроконтроллера и модуля управления мотором можно составить для строки.



Рисунок 34 - Пример теста для задачи 6

Решение задачи 6 на языке C++:

#include <iostream>

#include <vector>

#include <algorithm>

int main() {

int n;

std::cout << "Введите количество запросов: ";

std::cin >> n;

std::vector<int> results;

for (int i = 0; i < n; ++i) {

int a, b, x, y;

std::cout << "\nЗапрос " << (i + 1) << ":\n";

std::cout << "Введите количество микроконтроллеров типа A: ";

std::cin >> a;

std::cout << "Введите количество микроконтроллеров типа B: ";

std::cin >> b;

std::cout << "Введите количество модулей управления типа 1: ";

std::cin >> x;

std::cout << "Введите количество модулей управления типа 2: ";

std::cin >> y;

int b1 = std::min(b, x);

b -= b1;

x -= b1;

int a1 = std::min(a, x); // A + 1

a -= a1;

x -= a1;

int a2 = std::min(a, y); // A + 2

a -= a2;

y -= a2;

results.push\_back(b1 + a1 + a2);

}

std::cout << "\nМаксимальное количество работающих пар для каждого запроса:\n";

for (int r : results) {

std::cout << r << " ";

}

std::cout << "\n";

return 0;

}

Тесты:

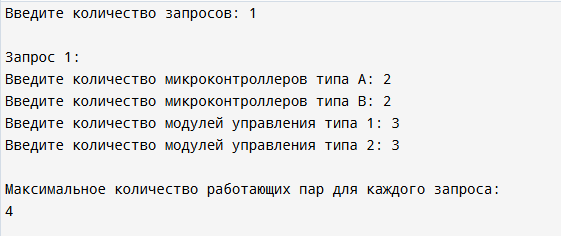


Рисунок 35 – Тест 1 для задачи 6

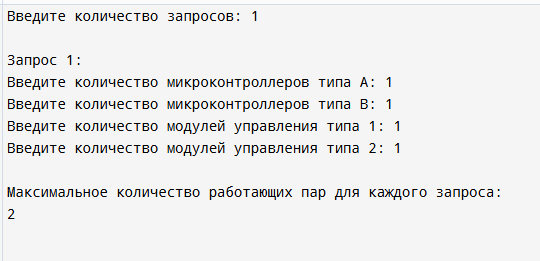


Рисунок 36 – Тест 2 для задачи 6

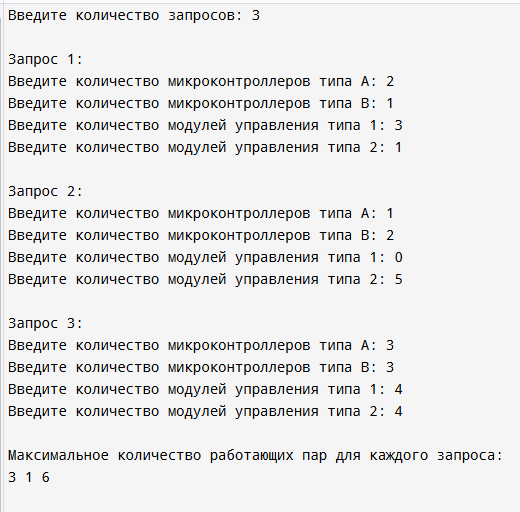


Рисунок 37 – Тест 3 для задачи 6

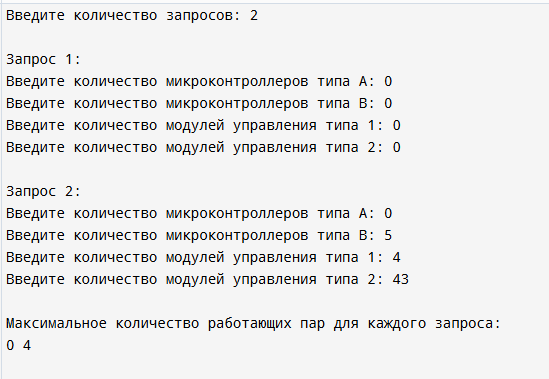


Рисунок 38 – Тест 4 для задачи 6

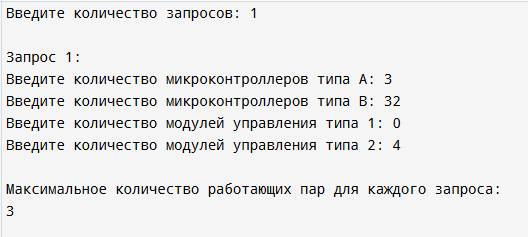


Рисунок 39 – Тест 5 для задачи 6