Міністерство освіти і науки України Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



Звіт

про виконання

**Лабораторних та практичних робіт № 5**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек»

***Виконав:***

студент групи ШІ-11 Боднар Роман Миколайович

Львів 2023

# Тема роботи:

# «Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек»

# Мета роботи:

Опанувати роботу з файлами. Навчитись користуватися вбудованими функціями та методами стандартної бібліотеки та файлами.

# Теоретичні відомості:

Джерела інформації:

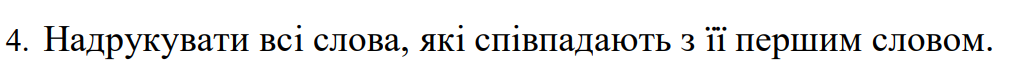
<https://www.tutorialspoint.com/reading-and-writing-binary-file-in-c-cplusplus>  
<https://www.dremendo.com/cpp-programming-tutorial/cpp-binary-file-handling>

Статус: Ознайомлений.

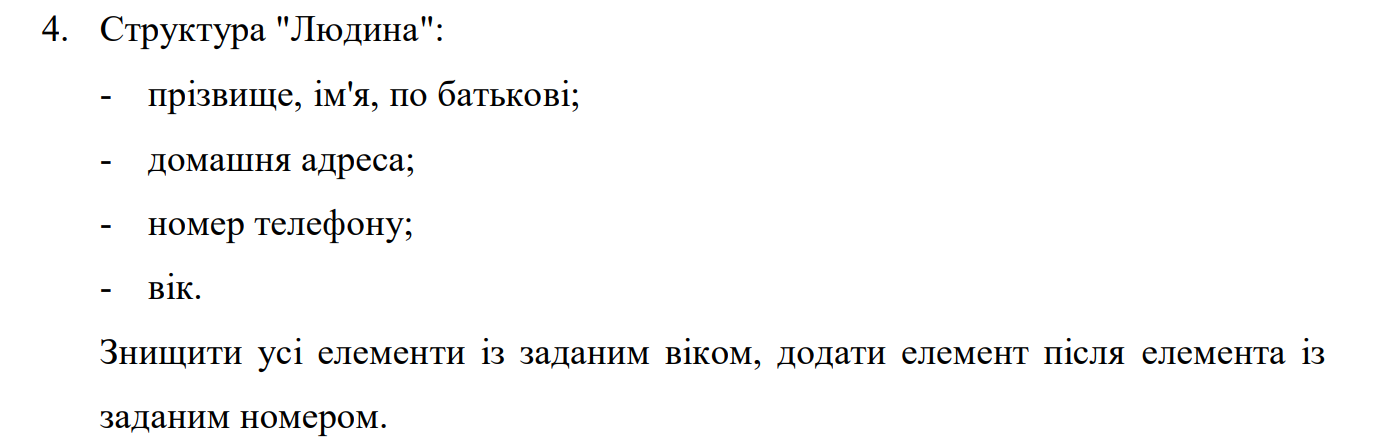
# Виконання роботи:

## Опрацювання завдання та вимог до програм та середовища:

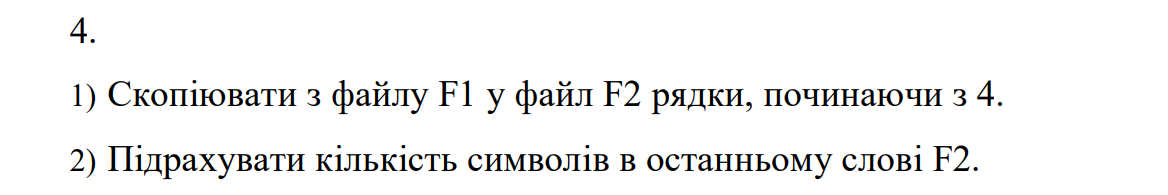
VNS Lab 6 – Variant 4:



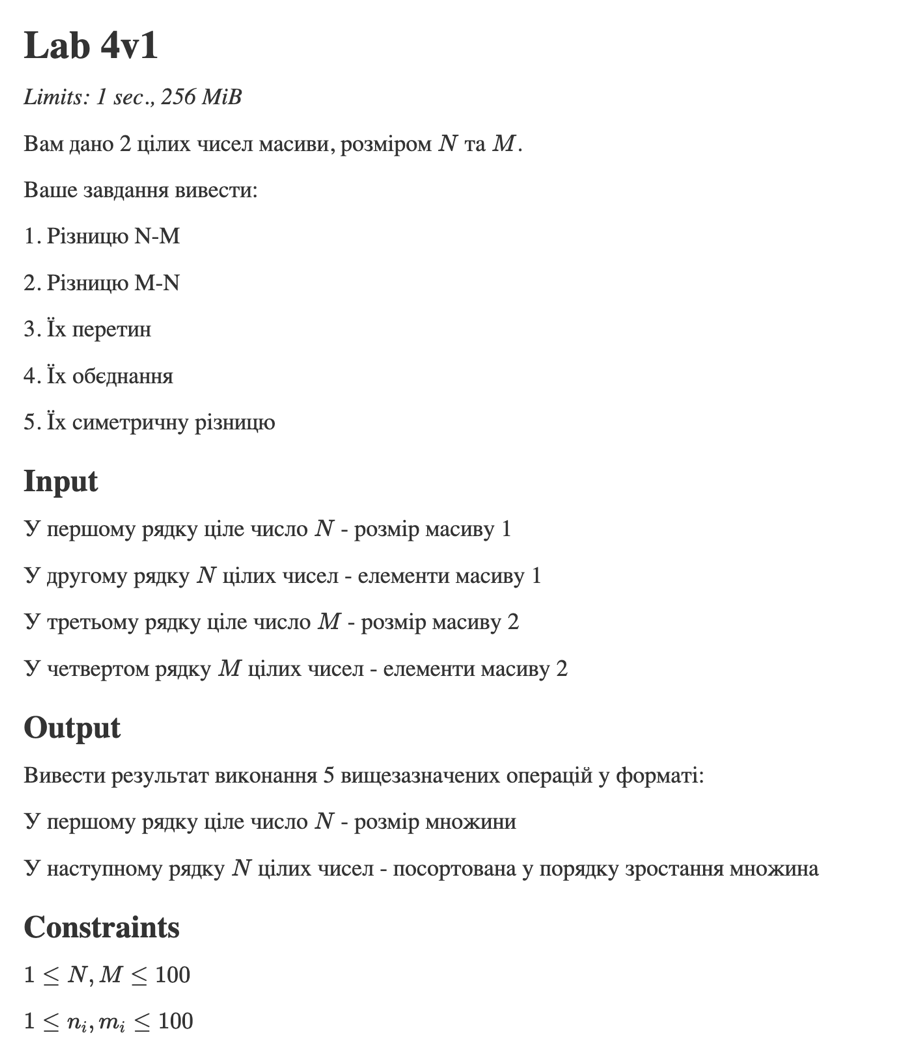
VNS Lab 8 – Variant 4:



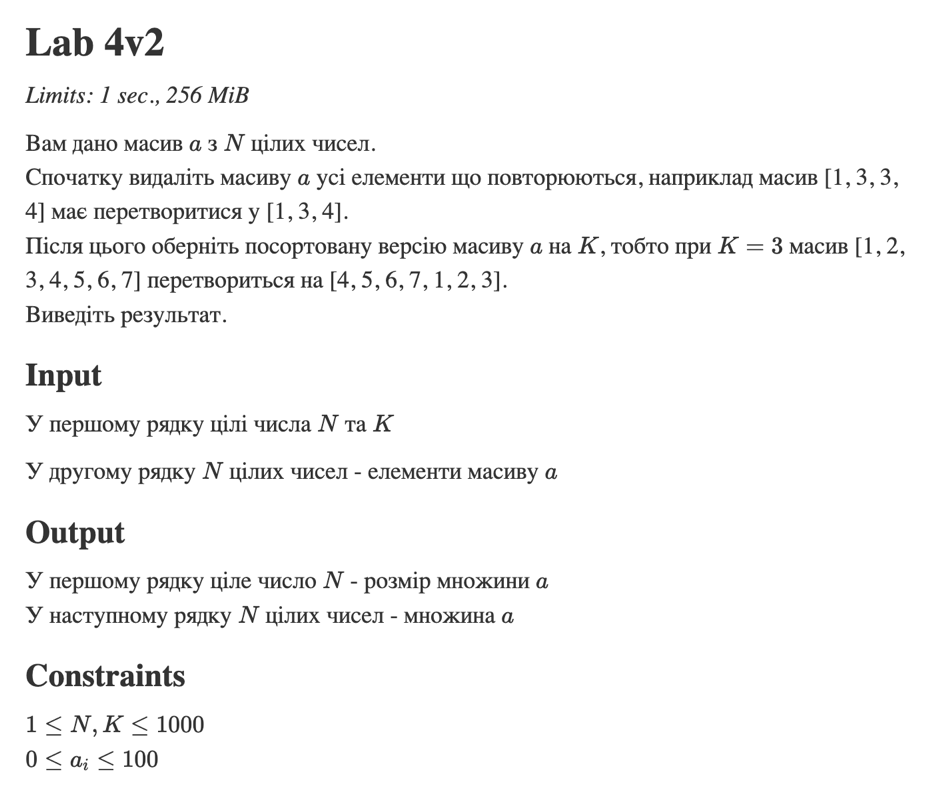
VNS Lab 9 – Variant 4:



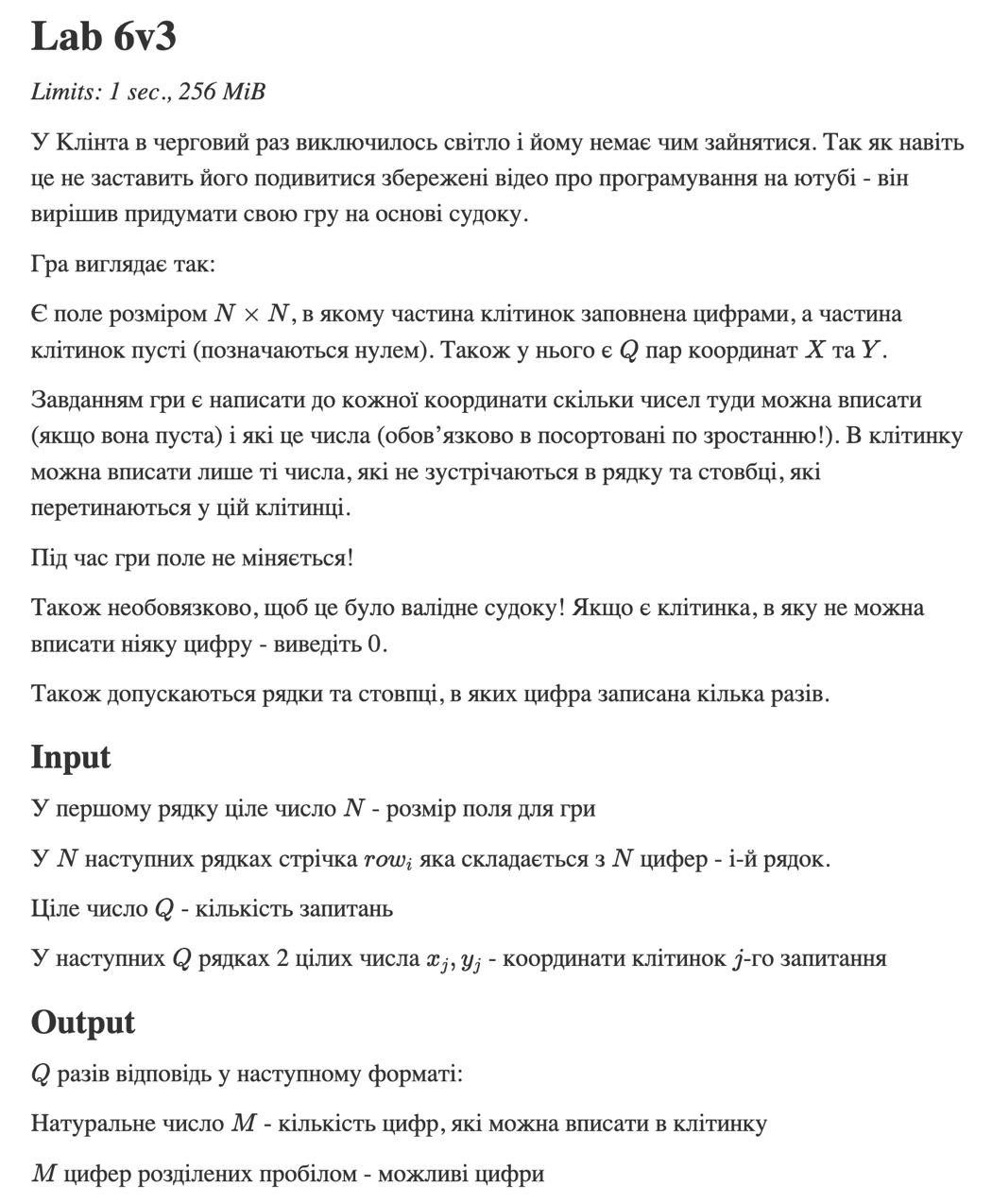
Algotester Lab 4.1:



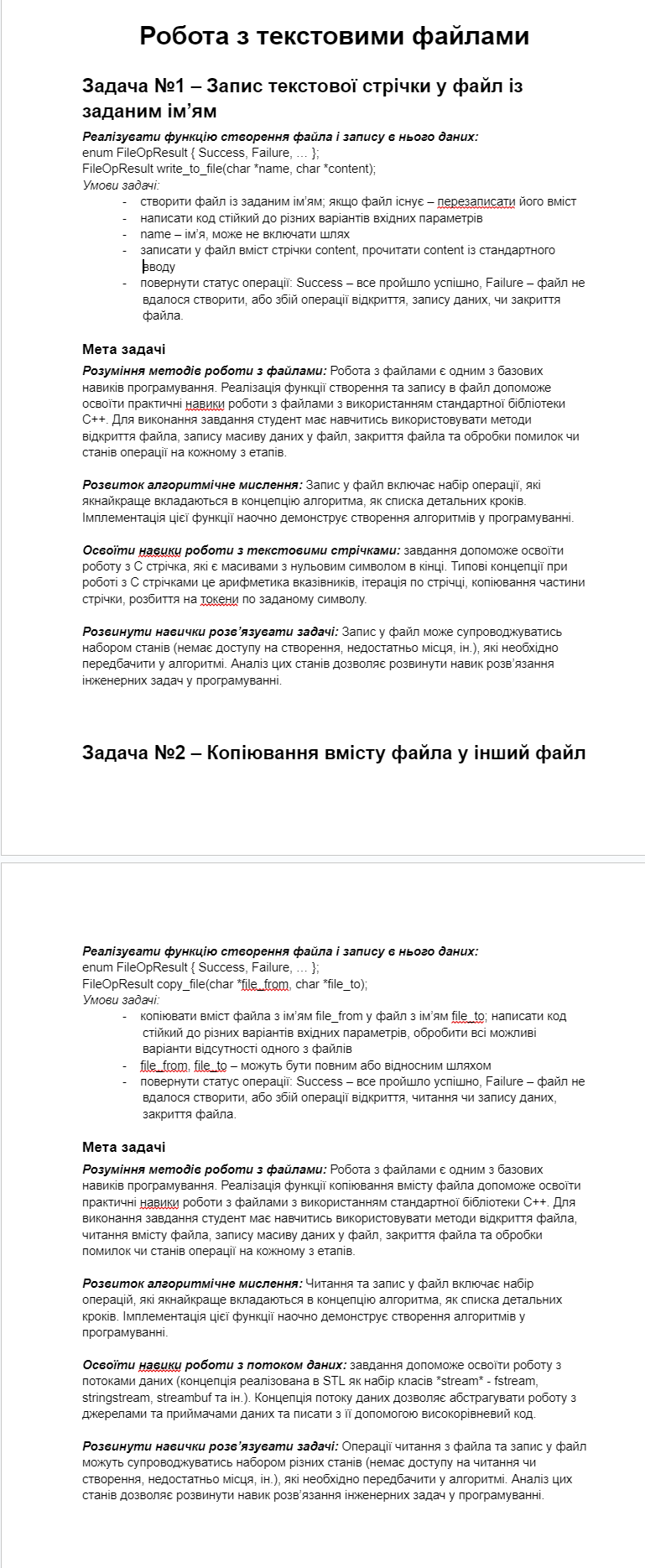
Algotester Lab 4.2:



Algotester Lab 6v3:

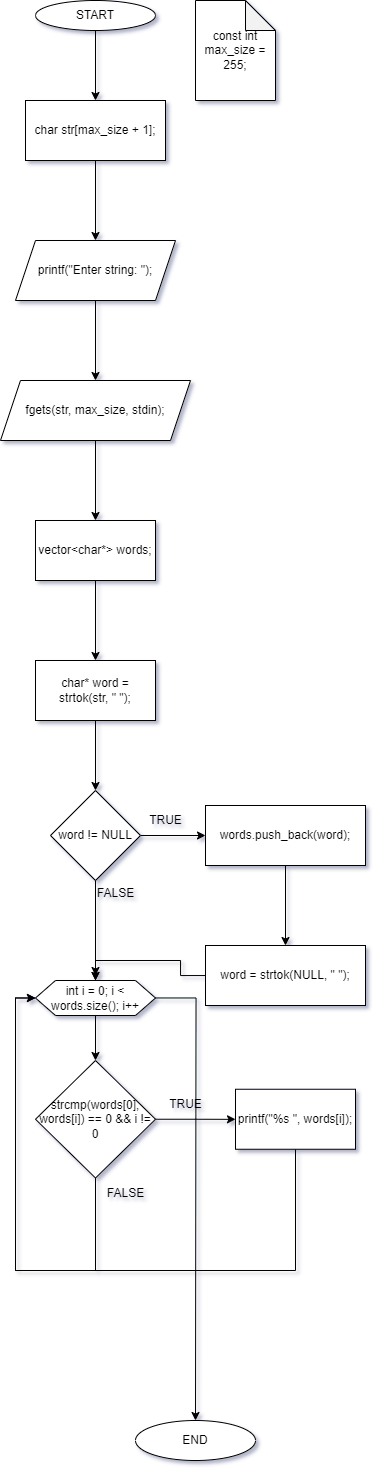


Class practice work:



## Дизайн та планована оцінка часу виконання завдань:

VNS Lab 6 – Variant 4:



Запланований час: 40 хв  
 Реальний час: 49 хв

## Код програм з посиланням на зовнішні ресурси:

VNS Lab 6 – Variant 4:

#include <iostream>  
#include <vector>  
#include <cstring>  
using namespace std;  
  
const int max\_size = 255;  
  
int main()  
{  
 // room for '\0'  
 char str[max\_size + 1];  
 printf("Enter string: ");  
 fgets(str, max\_size, stdin);  
  
 vector<char\*> words;  
 char\* word = strtok(str, " \n");  
 while (word != NULL)  
 {  
 words.push\_back(word);  
 word = strtok(NULL, " \n");  
 }  
  
 for (int i = 0; i < words.size(); i++)  
 if (strcmp(words[0], words[i]) == 0 && i != 0)  
 printf("%s ", words[i]);  
  
 return 0;  
}

VNS Lab 8 – Variant 4:

#include <iostream>  
#include <vector>  
#include <cstring>  
using namespace std;  
  
const int max\_size = 256;  
struct Human  
{  
 char name[max\_size + 1];  
 int age;  
 char home\_address[max\_size + 1];  
 char phone\_number[max\_size + 1];  
  
 Human()  
 {  
 strcpy(name, "");  
 age = 0;  
 strcpy(home\_address, "");  
 strcpy(phone\_number, "");  
 }  
  
 Human(const char \*name,  
 int age,  
 const char \*home\_address,  
 const char \*phone\_number)  
 {  
 strcpy(this->name, name);  
 this->age = age;  
 strcpy(this->home\_address, home\_address);  
 strcpy(this->phone\_number, phone\_number);  
 }  
};  
  
Human create\_human()  
{  
 Human human;  
 cout << "Enter name: ";  
 gets\_s(human.name, 255);  
 cout << "Enter age: ";  
 cin >> human.age;  
 getc(stdin); // remove \n from buffer (from previous cin)  
 cout << "Enter home address: ";  
 gets\_s(human.home\_address, 255);  
 cout << "Enter phone number: ";  
 gets\_s(human.phone\_number, 255);  
 return human;  
}  
  
int main()  
{  
 vector<Human> people;  
  
 FILE \*file = fopen("input.bin", "rb");  
 if (file == nullptr)  
 {  
 cerr << "File not found\nSeeding list\n";  
 people.emplace\_back("Roman", 18, "Lviv", "380931234567");  
 people.emplace\_back("Ivan", 19, "Kyiv", "380671234568");  
 }  
 else  
 {  
 Human human;  
 fread(&human, sizeof(Human), 1, file);  
 while (!feof(file))  
 {  
 people.push\_back(human);  
 fread(&human, sizeof(Human), 1, file);  
 }  
  
 fclose(file);  
 }  
  
 int choice;  
 do  
 {  
 cout << "1 - print people\n"  
 << "2 - add new person\n"  
 << "3 - delete person\n"  
 << "0 - exit\n"  
 << "Enter choice: ";  
  
 cin >> choice;  
  
 switch (choice)  
 {  
 case 1:  
 for (int i = 0; i < people.size(); i++)  
 cout << people[i].name << " " << people[i].age << " "  
 << people[i].home\_address << " "  
 << people[i].phone\_number << endl;  
 break;  
 case 2:  
 {  
 char phone\_number[max\_size];  
 cout << "Enter phone number of previous person: ";  
 getc(stdin); // remove \n from buffer (from previous cin)  
 gets\_s(phone\_number, 255);  
  
 for (int i = 0; i < people.size(); i++)  
 if (strcmp(people[i].phone\_number, phone\_number) == 0)  
 {  
 people.insert(people.begin() + i + 1, create\_human());  
 break;  
 }  
 break;  
 }  
 case 3:  
 {  
 int age;  
 cout << "Enter age of person to delete: ";  
 cin >> age;  
 for (int i = 0; i < people.size(); i++)  
 if (people[i].age == age){  
 people.erase(people.begin() + i);  
 i--;  
 }  
 break;  
 }  
 case 0:  
 break;  
 default:  
 cout << "Invalid choice" << endl;  
 break;  
 }  
 } while (choice != 0);  
  
 file = fopen("input.bin", "wb");  
 for (int i = 0; i < people.size(); i++)  
 fwrite(&people[i], sizeof(Human), 1, file);  
  
 fclose(file);  
  
 return 0;  
}

VNS Lab 9 – Variant 4:

#include <iostream>  
#include <string>  
#include <cstring>  
using namespace std;  
  
const int max\_line\_length = 256,  
 ignore\_lines = 4;  
  
int main()  
{  
 FILE \*f1 = fopen("f1.txt", "r+"),  
 \*f2 = fopen("f2.txt", "w+");  
  
 if (f1 == NULL || f2 == NULL) {  
 cout << "Error opening files" << endl;  
 return 1;  
 }  
  
 char line[max\_line\_length];  
  
 for (int i = 0; i < ignore\_lines && !feof(f1); i++) {  
 fgets(line, max\_line\_length, f1);  
 }  
  
 while (!feof(f1)) {  
 fgets(line, max\_line\_length, f1);  
 fputs(line, f2);  
 }  
  
 fclose(f1);  
 fclose(f2);  
  
 f2 = fopen("f2.txt", "r+");  
 if (f2 == NULL) {  
 cout << "Error opening file" << endl;  
 return 1;  
 }  
  
 while (!feof(f2)) fgets(line, max\_line\_length, f2);  
  
 // finds last space and returns pointer to it  
 char\* last\_word = strrchr(line, ' ') + 1;  
 cout << last\_word << endl << strlen(last\_word) << endl;  
  
 fclose(f2);  
  
 return 0;  
}

Algotester Lab 4.1:

#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
  
int main()  
{  
 int n;  
 cin >> n;  
  
 vector<int> mas(n);  
 for (int i = 0; i < n; ++i)  
 {  
 cin >> mas[i];  
 }  
  
 // Group elements into buckets  
 vector<vector<int> > buckets(3);  
 for (int i = 0; i < n; ++i)  
 {  
 buckets[mas[i] % 3].push\_back(mas[i]);  
 }  
  
 sort(buckets[0].begin(), buckets[0].end());  
 sort(buckets[1].begin(), buckets[1].end(), greater<int>());  
 sort(buckets[2].begin(), buckets[2].end());  
  
 mas.clear();  
 mas.insert(mas.end(), buckets[0].begin(), buckets[0].end());  
 mas.insert(mas.end(), buckets[1].begin(), buckets[1].end());  
 mas.insert(mas.end(), buckets[2].begin(), buckets[2].end());  
  
 auto last\_unique = unique(mas.begin(), mas.end());  
 mas.resize(distance(mas.begin(), last\_unique));  
  
 cout << mas.size() << endl;  
 for (int i = 0; i < mas.size(); ++i)  
 {  
 cout << mas[i] << " ";  
 }  
 cout << endl;  
  
 return 0;  
}

Algotester Lab 4.2:

#include <iostream>  
#include <vector>  
#include <algorithm>  
using namespace std;  
  
int main()  
{  
 int n;  
 cin >> n;  
  
 vector<int> mas(n);  
 for (int i = 0; i < n; ++i)  
 {  
 cin >> mas[i];  
 }  
  
 // Group elements into buckets  
 vector<vector<int>> buckets(3);  
 for (int i = 0; i < n; ++i)  
 {  
 buckets[mas[i] % 3].push\_back(mas[i]);  
 }  
  
 // Sort buckets  
 for (int i = 0; i < 3; ++i)  
 {  
 // Sort first bucket ascending, second bucket descending  
 if (i == 1)  
 {  
 sort(buckets[i].rbegin(), buckets[i].rend());  
 }  
 else  
 {  
 sort(buckets[i].begin(), buckets[i].end());  
 }  
 }  
  
 // Merge buckets back into `mas`  
 mas.clear();  
 for (const auto &bucket : buckets)  
 {  
 mas.insert(mas.end(), bucket.begin(), bucket.end());  
 }  
  
 // Remove duplicates  
 int i = 0;  
 int j = 1;  
 while (j < mas.size())  
 {  
 if (mas[i] == mas[j])  
 {  
 // Duplicate found, skip j  
 ++j;  
 }  
 else  
 {  
 // Keep element, advance both i and j  
 ++i;  
 mas[i] = mas[j];  
 ++j;  
 }  
 }  
 mas.resize(i + 1); // Shrink to new size  
  
 // Print results  
 cout << mas.size() << endl;  
 for (int i = 0; i < mas.size(); ++i)  
 {  
 cout << mas[i] << " ";  
 }  
 cout << endl;  
  
 return 0;  
}

Algotester Lab 6v3:

#include <iostream>  
#include <vector>  
#include <set>  
using namespace std;  
  
template<typename T>  
void print(const set<T>& \_set)  
{  
 cout << \_set.size() << endl;  
 for (auto &number : \_set)  
 cout << number << " ";  
 cout << endl;  
}  
  
int main()  
{  
 int n;  
 cin >> n;  
  
 vector<vector<int> > matrix(n, vector<int>(n));  
 for (int i = 0; i < n; ++i)  
 {  
 string line;  
 cin >> line;  
 for (int j = 0; j < n; ++j)  
 {  
 // cast digit char into respective int  
 matrix[i][j] = line[j] - '0';  
 }  
 }  
  
 int requests;  
 cin >> requests;  
  
 for (int i = 0; i < requests; ++i)  
 {  
 int x, y;  
 cin >> x >> y;  
  
 set<int> possibilities;  
 x--;  
 y--;  
  
 int current = matrix[x][y];  
 if (current) {  
 possibilities.insert(current);  
 print(possibilities);  
 continue;  
 }  
  
 for (int k = 1; k <= matrix.size(); ++k)  
 {  
 possibilities.insert(k);  
 }  
  
 for (int k = 0; k < matrix.size(); ++k)  
 {  
 possibilities.erase(matrix[x][k]);  
 possibilities.erase(matrix[k][y]);  
 }  
  
 print(possibilities);  
 }  
  
 return 0;  
}

Class practice work:

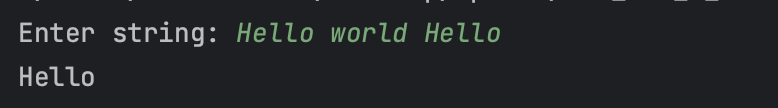
#include <iostream>  
using namespace std;  
  
enum FileOpResult  
{  
 *Success*,  
 *Failure*};  
  
FileOpResult write\_to\_file(char \*name, char \*content);  
FileOpResult copy\_file(char \*file\_from, char \*file\_to);  
  
int main()  
{  
 char name[256], content[256];  
 cout << "Enter file name: ";  
 gets\_s(name, 255);  
 cout << "Enter file content: ";  
 gets\_s(content, 255);  
  
 FileOpResult result1 = write\_to\_file(name, content);  
 if (result1 == FileOpResult::*Failure*)  
 {  
 cerr << "Error creating file" << endl;  
 return 1;  
 }  
 cout << "File created successfully" << endl;  
  
 char file\_from[256], file\_to[256];  
 cout << "Enter file to copy from: ";  
 gets\_s(file\_from, 255);  
 cout << "Enter file to copy to: ";  
 gets\_s(file\_to, 255);  
  
 FileOpResult result2 = copy\_file(file\_from, file\_to);  
 if (result2 == FileOpResult::*Failure*)  
 {  
 cerr << "Error copying file" << endl;  
 return 1;  
 }  
  
 return 0;  
}  
  
FileOpResult write\_to\_file(char \*name, char \*content)  
{  
 FILE \*f = fopen(name, "w+");  
 if (f == NULL)  
 {  
 return FileOpResult::*Failure*;  
 }  
 fputs(content, f);  
 fclose(f);  
 return FileOpResult::*Success*;  
}  
  
FileOpResult copy\_file(char \*file\_from, char \*file\_to)  
{  
 FILE \*src = fopen(file\_from, "r"),  
 \*dst = fopen(file\_to, "w+");  
  
 if (src == NULL || dst == NULL)  
 {  
 return FileOpResult::*Failure*;  
 }  
  
 char line[256];  
 while (!feof(src))  
 {  
 fgets(line, 256, src);  
 fputs(line, dst);  
 }  
  
 fclose(src);  
 fclose(dst);  
  
 return FileOpResult::*Success*;  
}

Self-practice:

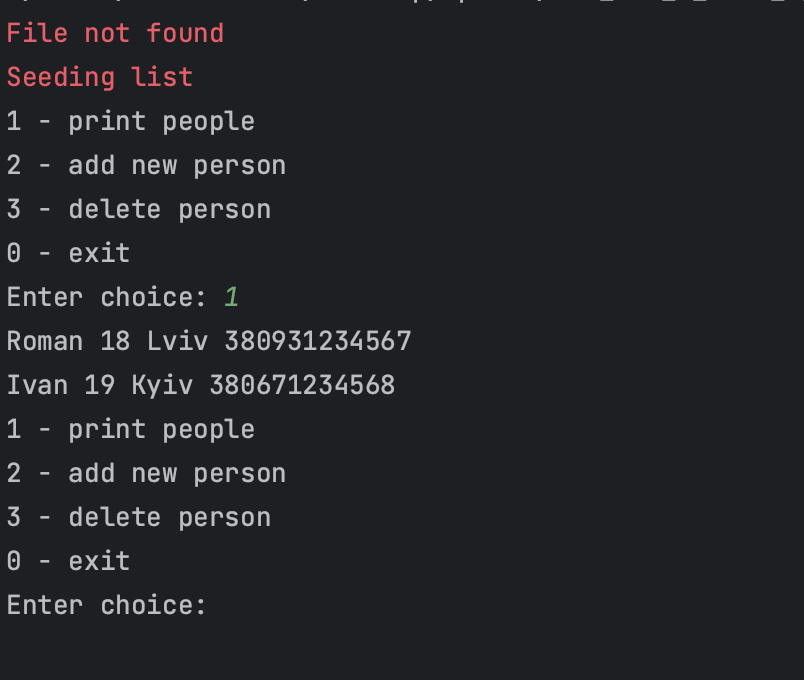
#include <iostream>  
#include <vector>  
#include <map>  
#include <set>  
#include <algorithm>  
using namespace std;  
  
// Lab 6v1  
int main() {  
 int count, min\_repetance;  
 cin >> count >> min\_repetance;  
  
 vector<string> arr(count);  
 map<string, int> counts;  
 set<char> c;  
  
 for (int i = 0; i < count; ++i) {  
 cin >> arr[i];  
 for (int j = 0; j < arr[i].size(); j++) {  
 arr[i][j] = tolower(arr[i][j]);  
 }  
 counts[arr[i]]++;  
 }  
   
 for (const auto& dict\_entry : counts)  
 if (dict\_entry.second >= min\_repetance)  
 for (int i = 0; i < dict\_entry.first.size(); ++i)  
 c.insert(dict\_entry.first[i]);  
  
 if (c.empty()) {  
 cout << "Empty!";  
 return 0;  
 }  
  
 cout << c.size() << endl;  
 for(auto it = c.rbegin(); it != c.rend(); ++it)  
 cout << \*it << " ";  
  
 return 0;  
}

**4. Результати виконання завдань, тестування та фактично витрачений час:**

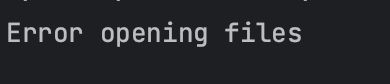
VNS Lab 6 – Variant 4:



VNS Lab 8 – Variant 4:

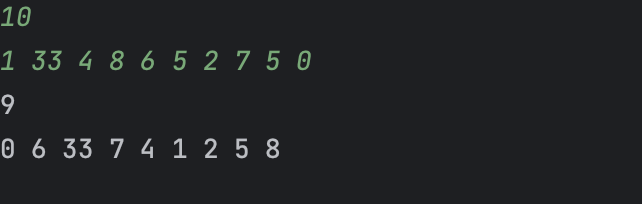


VNS Lab 9 – Variant 4:

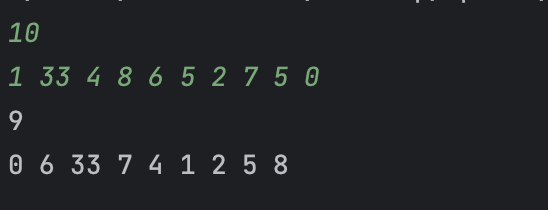


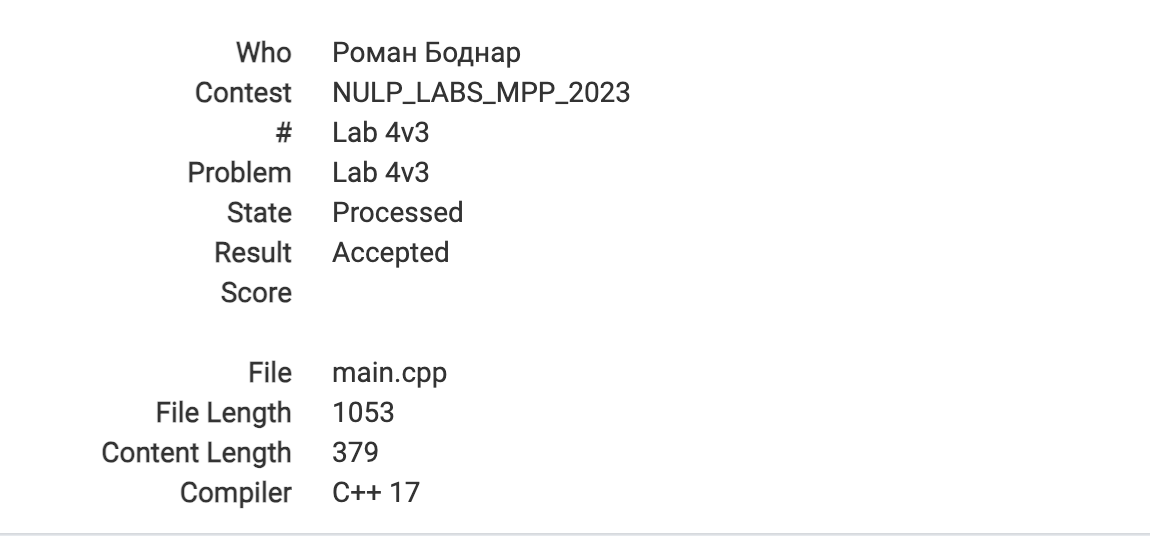


Algotester Lab 4.1:

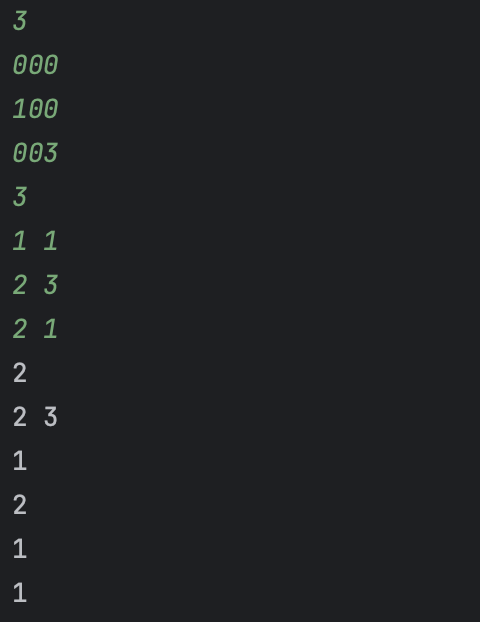


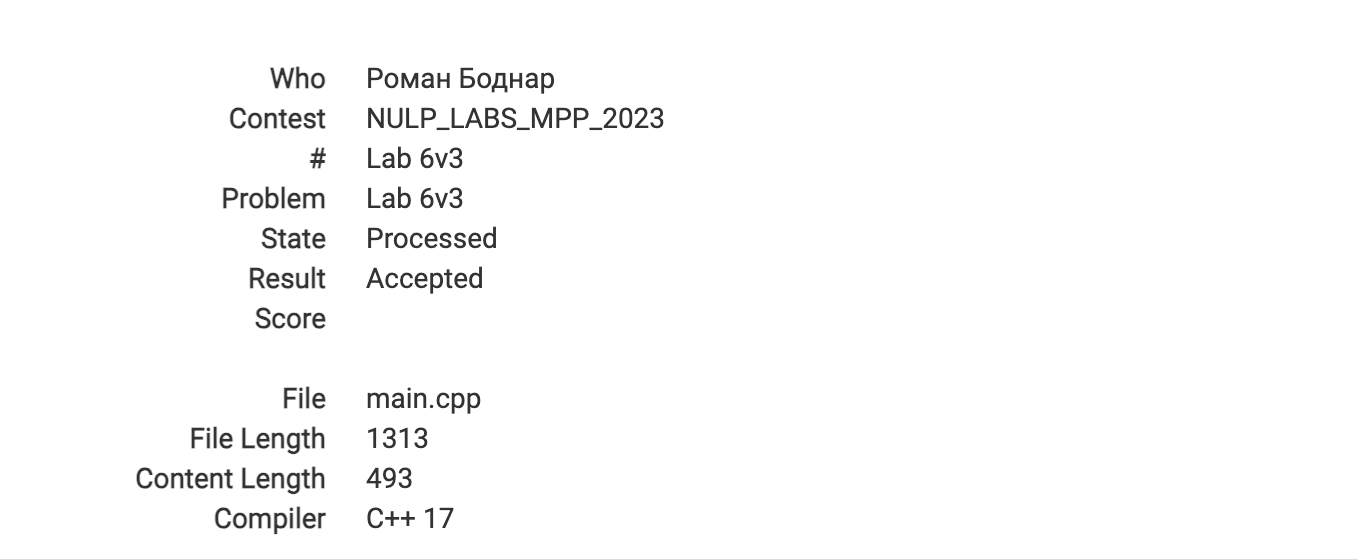
Algotester Lab 4.2:

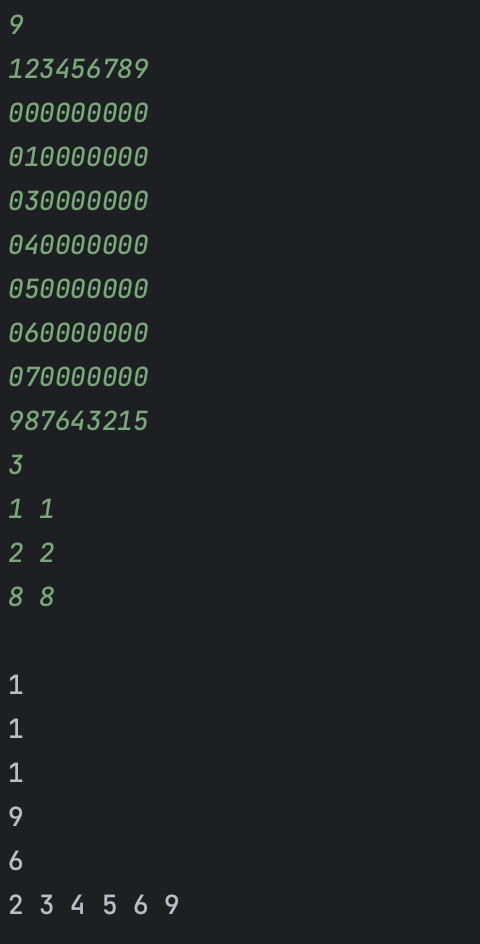




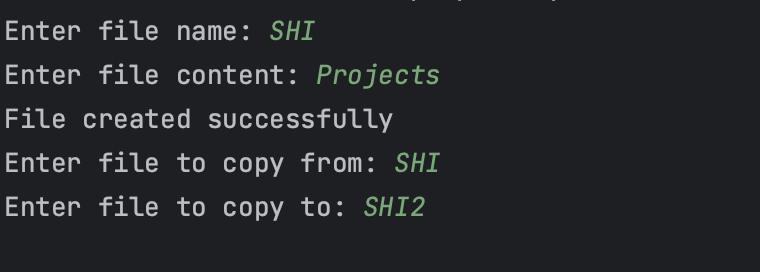
Algotester Lab 6v3:

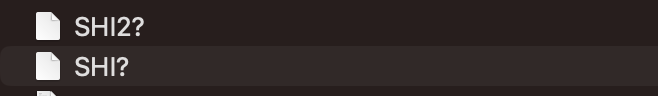






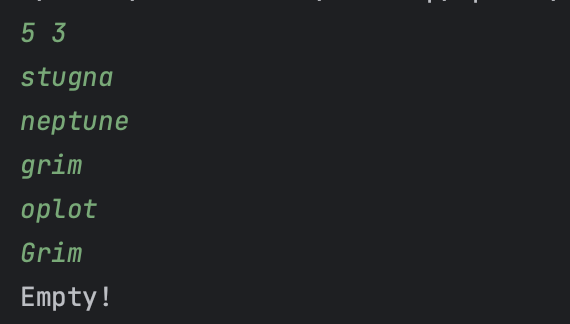
Class practice work:





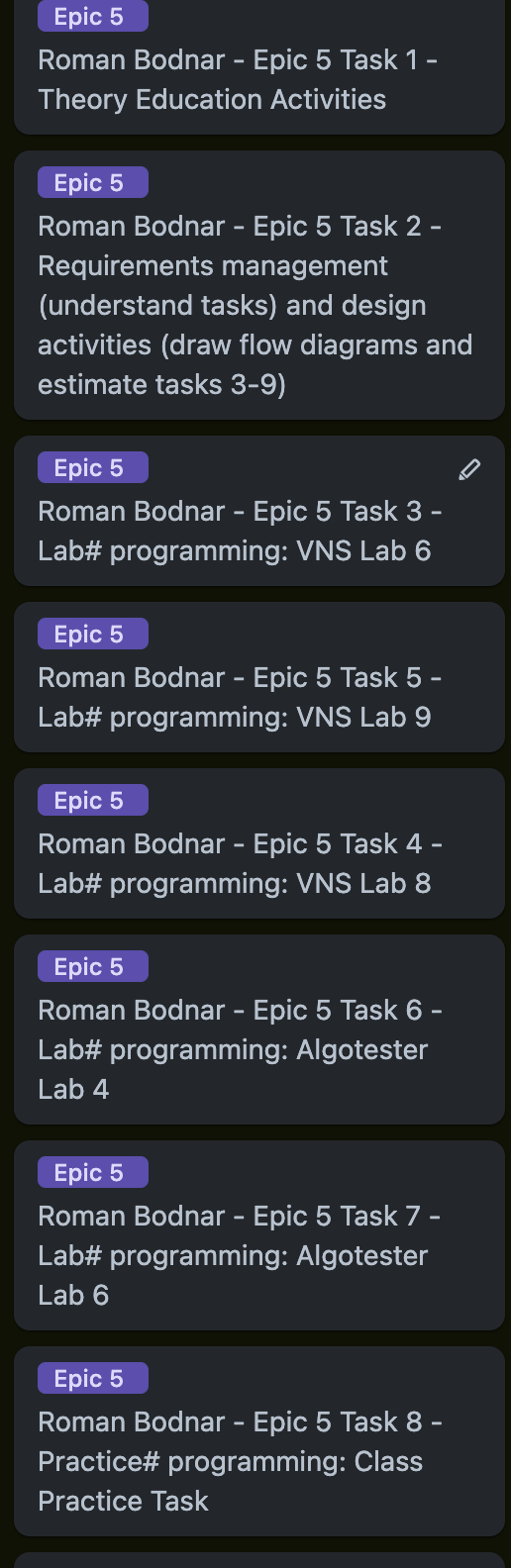
Self-practice:

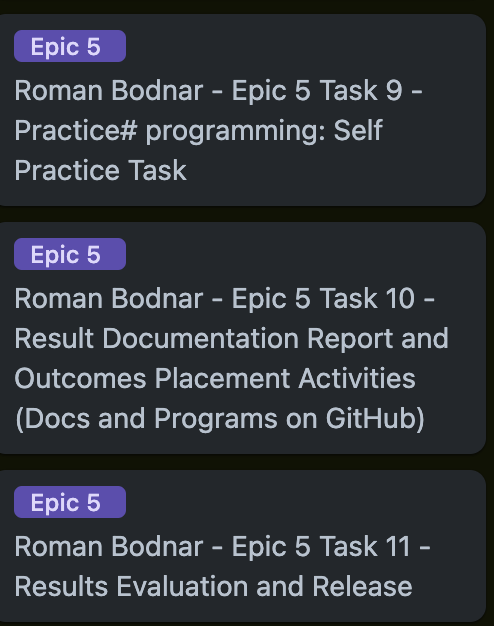




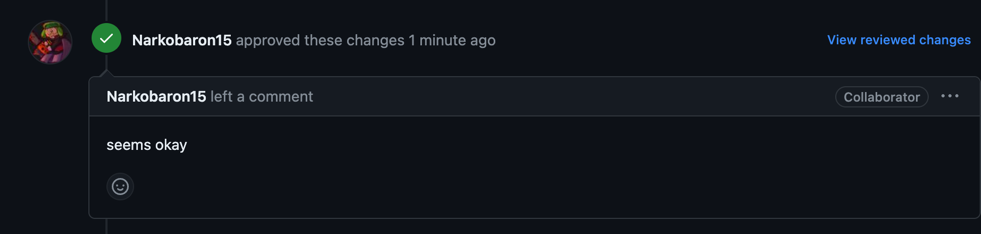
PR: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1178>

**Скрін трелло в Review по Епік 5:**





Коментарі PR:



# Висновки:

В даній лабораторній роботі я опанував теоретичні і практичні знання файлів і бінарних файлів, стандартних бібліотек, символів і рядкових змінних.