Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт №6 *з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур»

***Виконала:***

студент групи ШІ-12

Степова Марія Андріївна

# **Тема роботи:**

Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур

# **Мета роботи:**

Розуміння роботи та застосування різних структур даних для ефективного зберігання та управління інформацією, розуміння алгоритмів, які працюють з динамічними структурами, для виконання пошуку, сортування, обходу та інших операцій над даними.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: **Динамічні структури**.

1. Індивідуальний план опрацювання теорії:

* Тема №1: **Динамічні структури**.
  + Джерела Інформації
    - <https://ppt-online.org/889534>
    - <https://studfile.net/preview/3908535/>
    - <https://youtu.be/mdfUG6Siro0>
    - <https://www.youtube.com/watch?v=VTwgrdh_D34&ab_channel=%D0%9E%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80%D0%86%D0%B2%D0%B0%D0%BD%D0%BE%D0%B2%D0%B8%D1%87%D0%9C%D0%B0%D1%80%D1%87%D0%B5%D0%BD%D0%BA%D0%BE>
    - <https://www.geeksforgeeks.org/what-is-linked-list/>
  + Що опрацьовано:
    - Завдяки опрацьованому матеріалу я знаю, що динамічні структури даних - це типи структур даних, які дозволяють гнучко керувати пам'яттю та змінювати розмір даних у виконанні програми, списки - це колекція елементів, де кожен елемент має посилання на наступний елемент у списку, стеки - це структура даних де останній елемент, який був доданий, буде першим, що буде вилучено, черги - це структура даних, де перший елемент, який був доданий, буде першим, що буде вилучено, дерева складаються з вузлів, де кожен вузол може мати декілька дочірніх вузлів. Дерева можуть мати різні форми, такі як бінарні дерева.
  + Статус: **Ознайомлений**
  + Початок опрацювання теми: **28.11.2023**
  + Звершення опрацювання теми: **08.12.2023**

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 **VNS Lab 10 - Task 1-11**

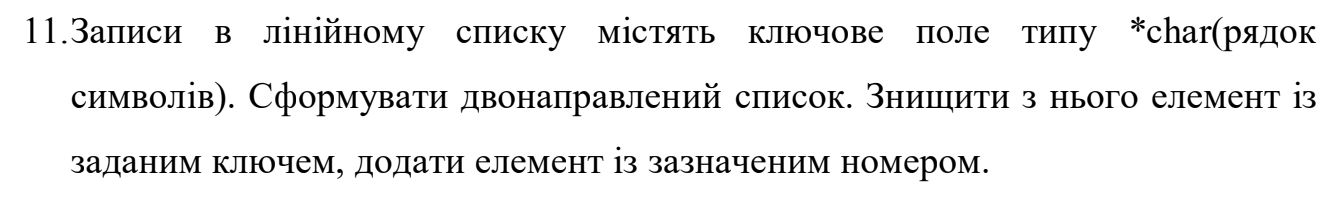
* Деталі завдання
* 

Рисунок 1. Скриншот завдання

Завдання №2 **Algotester Lab 5**

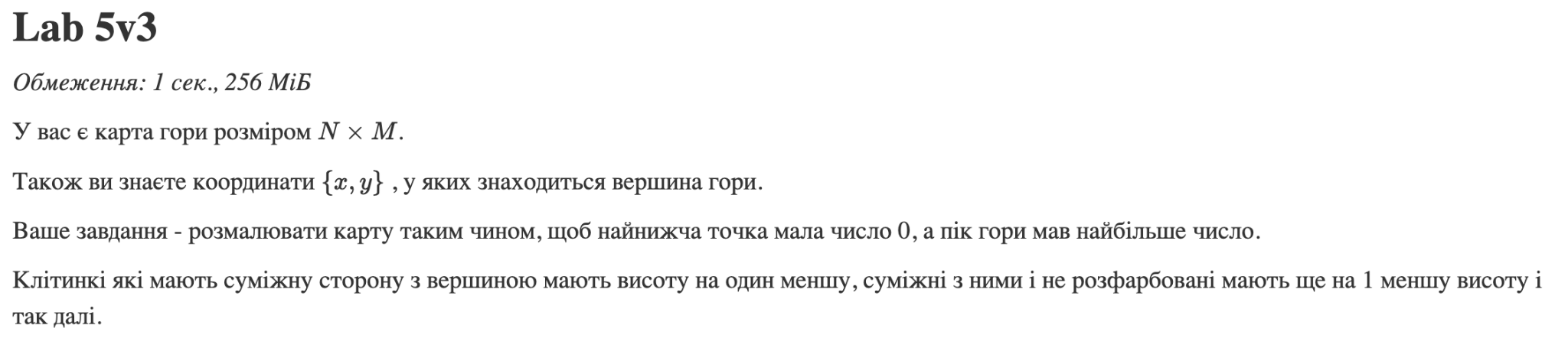
* Деталі завдання
* 

Рисунок 2. Скриншот завдання

Завдання №3 **Algotester Lab 7-8**

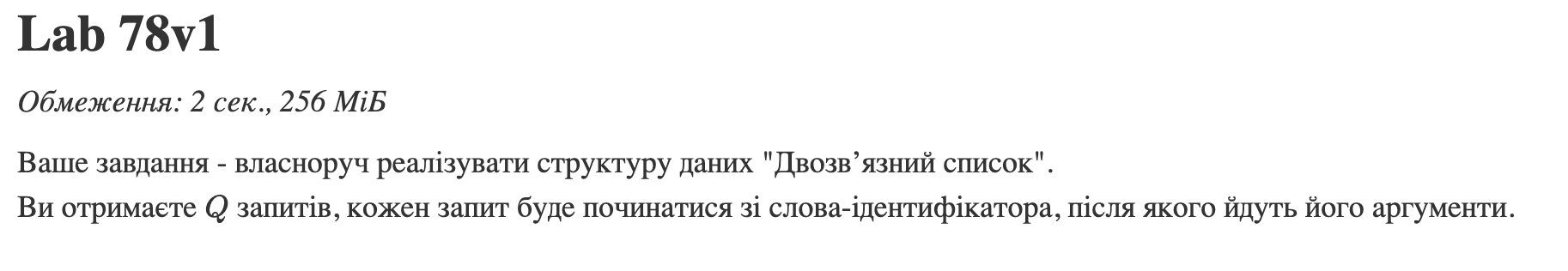
* Деталі завдання
* 

Рисунок 3. Скриншот завдання

Завдання №4 **Class Practice Work**

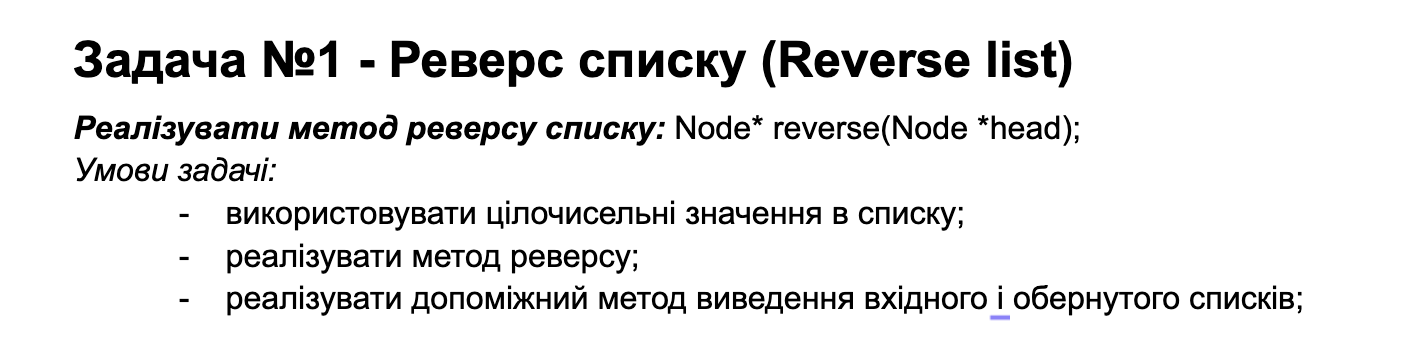
* Деталі завдання
* 

Рисунок 4. Скриншот завдання

Завдання №5 **Class Practice Work**

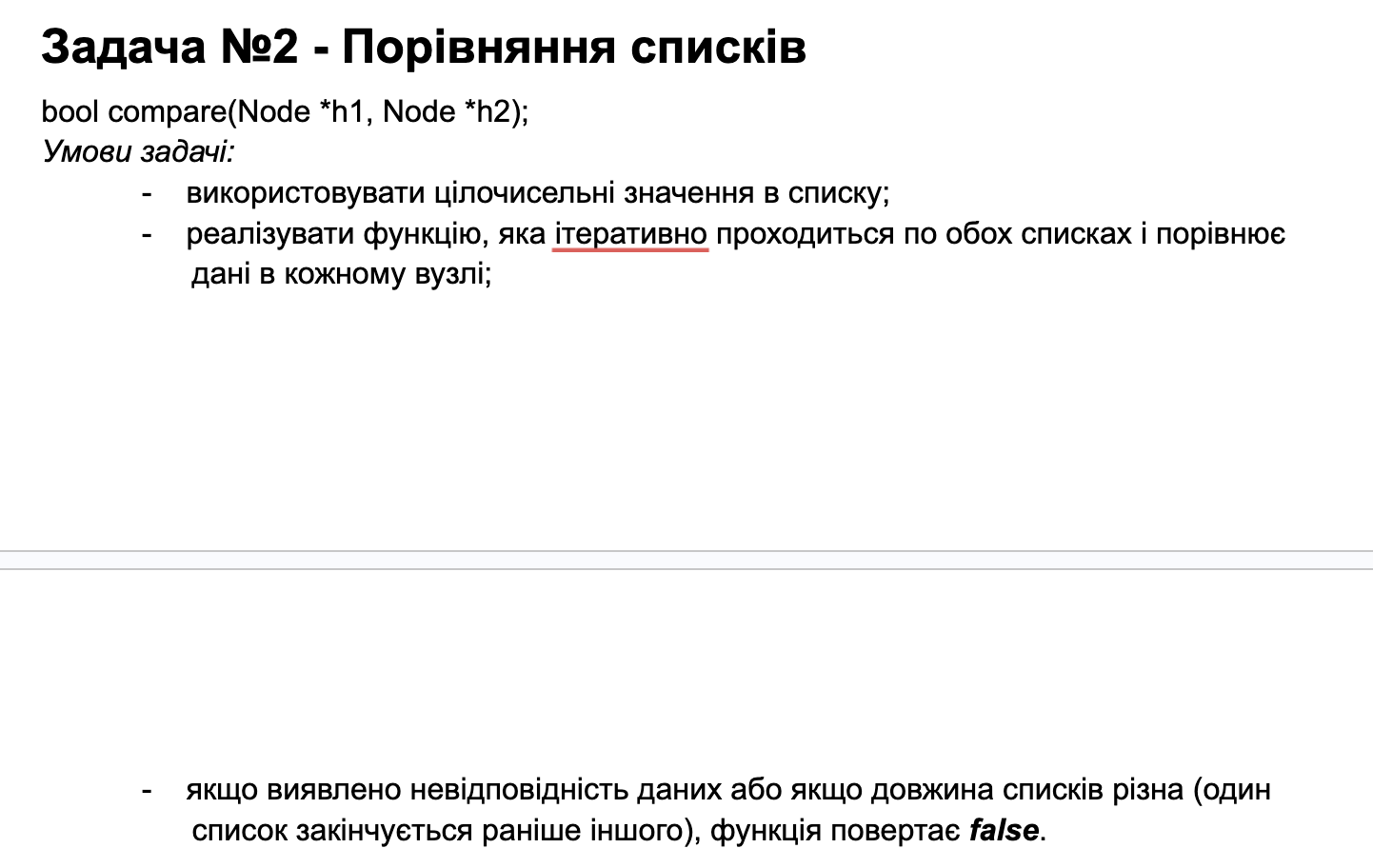
* Деталі завдання
* 

Рисунок 5. Скриншот завдання

Завдання №6 **Class Practice Work**

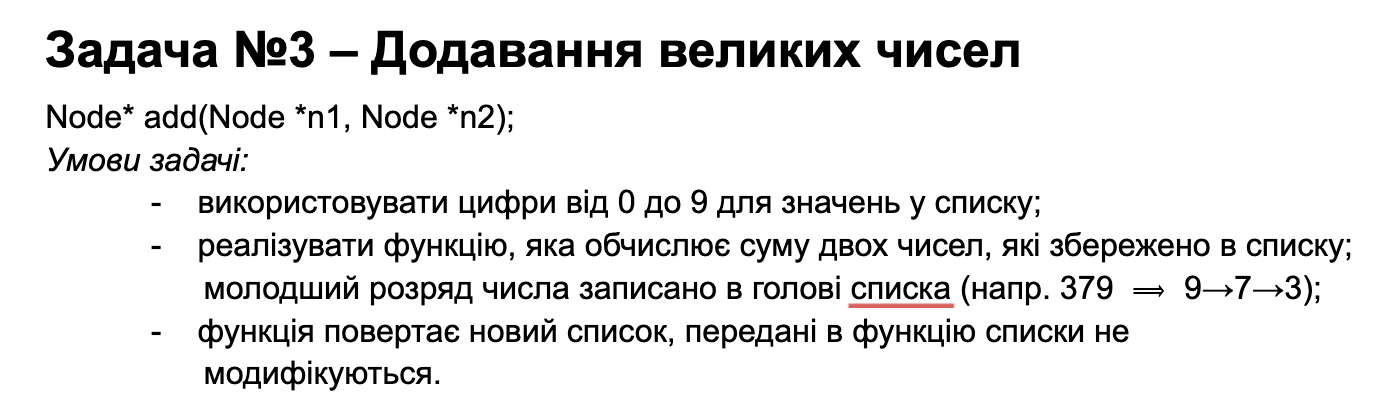
* Деталі завдання
* 

Рисунок 6. Скриншот завдання

Завдання №7 **Self-Practice Work**

* Деталі завдання



Рисунок 7. Скриншот завдання

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №2 **Algotester Lab 5**

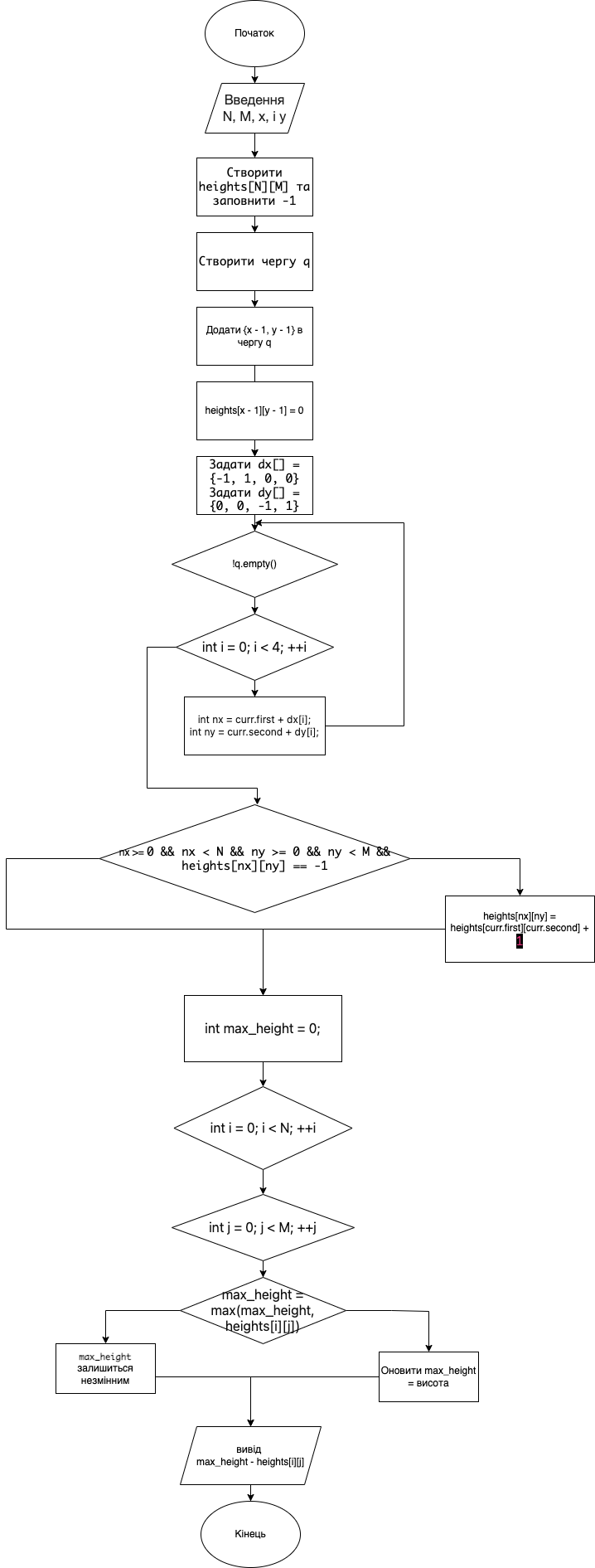
* Блок-схема
* 

Рисунок 8. Блок-схема до завдання

* Планований час на реалізацію: **2 години**

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання №1 **VNS Lab 10 - Task 1-11**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

#include <fstream>

#include <string>

using *namespace* std;

*struct* Node {

string data;

Node\* next;

Node\* prev;

Node(const string& *data*) : data(*data*), next(nullptr), prev(nullptr) {}

};

*class* DoublyLinkedList {

*public:*

Node\* head;

Node\* tail;

DoublyLinkedList() : head(nullptr), tail(nullptr) {}

*void* push\_back(const string& *value*) {

Node\* newNode = new Node(*value*);

if (tail == nullptr) {

head = tail = newNode;

} else {

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

}

*void* printList() {

Node\* current = head;

if (current == nullptr) {

cout << "Список порожній" << endl;

return;

}

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

*void* deleteByKey(const string& *key*) {

Node\* current = head;

while (current != nullptr) {

Node\* nextNode = current->next;

if (current->data == *key*) {

if (current->prev != nullptr) {

current->prev->next = current->next;

} else {

head = current->next;

if (head != nullptr) {

head->prev = nullptr;

}

}

if (current->next != nullptr) {

current->next->prev = current->prev;

} else {

tail = current->prev;

if (tail != nullptr) {

tail->next = nullptr;

}

}

delete current;

}

current = nextNode;

}

}

*void* addAtPosition(const string& *data*, *int* *position*) {

Node\* newNode = new Node(*data*);

if (*position* == 0) {

newNode->next = head;

if (head != nullptr) {

head->prev = newNode;

}

head = newNode;

if (tail == nullptr) {

tail = newNode;

}

} else {

Node\* current = head;

for (*int* i = 0; i < *position*; ++i) {

if (current == nullptr) {

cout << "Помилка: Зазначена позиція перевищує довжину списку" << endl;

return;

}

current = current->next;

}

if (current != nullptr) {

newNode->prev = current->prev;

newNode->next = current;

if (current->prev != nullptr) {

current->prev->next = newNode;

} else {

head = newNode;

}

current->prev = newNode;

}

}

}

*void* writeTo(const string& *file*) {

ofstream outFile(*file*);

Node\* current = head;

while (current != nullptr) {

outFile << current->data << endl;

current = current->next;

}

outFile.close();

}

*void* readFrom(const string& *file*) {

ifstream inFile(*file*);

string line;

while (getline(inFile, line)) {

push\_back(line);

}

inFile.close();

}

*void* clearList() {

Node\* current = head;

while (current != nullptr) {

Node\* next = current->next;

delete current;

current = next;

}

head = tail = nullptr;

}

~DoublyLinkedList() {

clearList();

}

};

*int* main() {

DoublyLinkedList myList;

myList.push\_back("Список");

myList.push\_back("рядків");

myList.push\_back("для");

myList.push\_back("запису");

cout << "Початковий список:" << endl;

myList.printList();

myList.deleteByKey("для");

cout << "Список після видалення:" << endl;

myList.printList();

myList.addAtPosition("важливих", 2);

cout << "Список після додавання:" << endl;

myList.printList();

myList.writeTo("myList.txt");

myList.clearList();

cout << "Список після очищення:" << endl;

myList.printList();

myList.readFrom("myList.txt");

cout << "Список після відновлення:" << endl;

myList.printList();

return 0;

}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-f44bd192b51ebd086b8460bdcbd837d2ab3dca174b19baf5caa607a5707d0687>

Завдання №2 **Algotester Lab 5**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

#include <vector>

#include <queue>

using *namespace* std;

*int* main() {

*int* N, M, x, y;

cin >> N >> M >> x >> y;

vector<vector<*int*>> heights(N, vector<*int*>(M, -1));

queue<pair<*int*, *int*>> q;

q.push({x - 1, y - 1});

heights[x - 1][y - 1] = 0;

*int* dx[] = {-1, 1, 0, 0};

*int* dy[] = {0, 0, -1, 1};

while (!q.empty()) {

pair<*int*, *int*> curr = q.front();

q.pop();

for (*int* i = 0; i < 4; ++i) {

*int* nx = curr.first + dx[i];

*int* ny = curr.second + dy[i];

if (nx >= 0 && nx < N && ny >= 0 && ny < M && heights[nx][ny] == -1) {

heights[nx][ny] = heights[curr.first][curr.second] + 1;

q.push({nx, ny});

}

}

}

*int* max\_height = 0;

for (*int* i = 0; i < N; ++i) {

for (*int* j = 0; j < M; ++j) {

max\_height = max(max\_height, heights[i][j]);

}

}

for (*int* i = 0; i < N; ++i) {

for (*int* j = 0; j < M; ++j) {

cout << max\_height - heights[i][j] << " ";

}

cout << "\n";

}

return 0;

}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-47d2527336da77704338d706975084c3fd20201691e53e6837b4b81661f861a2>

Завдання №3 **Algotester Lab 7-8**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

*class* Node {

*public:*

*int* data;

Node\* prev;

Node\* next;

Node(*int* *data*) : data(*data*), prev(nullptr), next(nullptr) {}

};

*class* DoublyLinkedList {

*private:*

Node\* head;

Node\* tail;

*int* size;

*public:*

DoublyLinkedList() : head(nullptr), tail(nullptr), size(0) {}

*void* insert(*int* *index*, *int* *n*, *int*\* *arr*) {

if (*index* < 0 || *index* > size) {

std::cout << "Invalid index for insertion." << std::endl;

return;

}

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

Node\* newNode = new Node(*arr*[i]);

if (head == nullptr) {

head = newNode;

tail = newNode;

} else if (*index* == 0) {

newNode->next = head;

head->prev = newNode;

head = newNode;

} else if (*index* == size) {

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

} else {

Node\* temp = head;

for (*int* j = 0; j < *index* - 1; ++j) {

temp = temp->next;

}

newNode->next = temp->next;

newNode->prev = temp;

temp->next->prev = newNode;

temp->next = newNode;

}

++size;

++*index*;

}

}

*void* erase(*int* *index*, *int* *n*) {

if (*index* < 0 || *index* >= size || *n* <= 0) {

std::cout << "Invalid index or number of elements to erase." << std::endl;

return;

}

while (*n* > 0 && *index* < size) {

Node\* temp;

if (*index* == 0) {

temp = head;

head = head->next;

if (head != nullptr) {

head->prev = nullptr;

}

} else if (*index* == size - 1) {

temp = tail;

tail = tail->prev;

if (tail != nullptr) {

tail->next = nullptr;

}

} else {

Node\* curr = head;

for (*int* i = 0; i < *index*; ++i) {

curr = curr->next;

}

temp = curr;

curr->prev->next = curr->next;

curr->next->prev = curr->prev;

}

delete temp;

--size;

--*n*;

}

}

*int* getSize() {

return size;

}

*int* get(*int* *index*) {

if (*index* < 0 || *index* >= size) {

std::cout << "Invalid index to get element." << std::endl;

return -1; // Return a default value indicating an error

}

Node\* temp = head;

for (*int* i = 0; i < *index*; ++i) {

temp = temp->next;

}

return temp->data;

}

*void* set(*int* *index*, *int* *value*) {

if (*index* < 0 || *index* >= size) {

std::cout << "Invalid index to set element." << std::endl;

return;

}

Node\* temp = head;

for (*int* i = 0; i < *index*; ++i) {

temp = temp->next;

}

temp->data = *value*;

}

friend std::ostream& operator<<(std::ostream& *os*, DoublyLinkedList& *list*) {

Node\* temp = *list*.head;

while (temp != nullptr) {

*os* << temp->data << " ";

temp = temp->next;

}

return *os*;

}

~DoublyLinkedList() {

Node\* temp = head;

while (temp != nullptr) {

Node\* next = temp->next;

delete temp;

temp = next;

}

head = nullptr;

tail = nullptr;

size = 0;

}

};

*int* main() {

*int* Q;

std::cin >> Q;

DoublyLinkedList list;

while (Q--) {

std::string query;

std::cin >> query;

if (query == "insert") {

*int* index, n;

std::cin >> index >> n;

*int*\* arr = new *int*[n];

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

std::cin >> arr[i];

}

list.insert(index, n, arr);

delete[] arr;

} else if (query == "erase") {

*int* index, n;

std::cin >> index >> n;

list.erase(index, n);

} else if (query == "size") {

std::cout << list.getSize() << std::endl;

} else if (query == "get") {

*int* index;

std::cin >> index;

std::cout << list.get(index) << std::endl;

} else if (query == "set") {

*int* index, value;

std::cin >> index >> value;

list.set(index, value);

} else if (query == "print") {

std::cout << list << std::endl;

} else {

std::cout << "Invalid query." << std::endl;

}

}

return 0;

}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-73fb2be37e14641e7992be0209c8d316e2ed792e9f34d99b765c2249ad5d9623>

Завдання №4 **Class Practice Work**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

*struct* Node {

*int* data;

Node\* next;

Node(*int* *val*) : data(*val*), next(nullptr) {}

};

*class* LinkedList {

*public:*

Node\* reverse(Node\* *head*) {

Node\* prev = nullptr;

Node\* current = *head*;

Node\* next = nullptr;

while (current != nullptr) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

return prev; // новий початок списку

}

*void* printList(Node\* *node*) {

while (*node* != nullptr) {

std::cout << *node*->data << " ";

*node* = *node*->next;

}

std::cout << std::endl;

}

};

*int* main() {

using *namespace* std;

LinkedList list;

Node\* head = new Node(1);

head->next = new Node(2);

head->next->next = new Node(3);

head->next->next->next = new Node(4);

cout << "Оригінальний список: ";

list.printList(head);

head = list.reverse(head);

cout << "Обернутий список: ";

list.printList(head);

return 0;

}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-af9845d5015d2e964945a0bb6fe9f94930e7a595742de25dcf972aaeeba5ccd9>

Завдання №5 **Class Practice Work**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

*struct* Node {

*int* data;

Node\* next;

Node(*int* *val*) : data(*val*), next(nullptr) {}

};

*class* LinkedList {

*public:*

*bool* compare(Node\* *h1*, Node\* *h2*) {

while (*h1* != nullptr && *h2* != nullptr) {

if (*h1*->data != *h2*->data) {

return false;

}

*h1* = *h1*->next;

*h2* = *h2*->next;

}

return (*h1* == nullptr && *h2* == nullptr);

}

};

*int* main() {

using *namespace* std;

LinkedList list;

Node\* head1 = new Node(1);

head1->next = new Node(2);

head1->next->next = new Node(3);

Node\* head2 = new Node(1);

head2->next = new Node(2);

head2->next->next = new Node(3);

cout << "Результат порівняння списків: ";

if (list.compare(head1, head2)) {

cout << "Списки ідентичні.\n";

} else {

cout << "Списки не ідентичні.\n";

}

return 0;

}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-b8cf01d90ec80b8a521d6dfb6f43d16a94da7aad4273e659c117484bce02f445>

Завдання №6 **Class Practice Work**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

*struct* Node {

*int* data;

Node\* next;

Node(*int* *val*) : data(*val*), next(nullptr) {}

};

Node\* add(Node\* *n1*, Node\* *n2*) {

Node\* result = nullptr;

Node\* current = nullptr;

*int* carry = 0;

while (*n1* != nullptr n2 != nullptr carry != 0) {

*int* sum = carry;

if (*n1* != nullptr) {

sum += *n1*->data;

*n1* = *n1*->next;

}

if (*n2* != nullptr) {

sum += *n2*->data;

*n2* = *n2*->next;

}

carry = sum / 10;

sum = sum % 10;

Node\* newNode = new Node(sum);

if (result == nullptr) {

result = newNode;

current = newNode;

} else {

current->next = newNode;

current = newNode;

}

}

return result;

}

*void* printList(Node\* *node*) {

while (*node* != nullptr) {

std::cout << *node*->data << " ";

*node* = *node*->next;

}

std::cout << std::endl;

}

*int* main() {

using *namespace* std;

// Приклади чисел

Node\* num1 = new Node(3);

num1->next = new Node(7);

num1->next->next = new Node(9);

Node\* num2 = new Node(4);

num2->next = new Node(5);

num2->next->next = new Node(2);

cout << "Число 1: ";

printList(num1);

cout << "Число 2: ";

printList(num2);

Node\* result = add(num1, num2);

cout << "Сума: ";

printList(result);

return 0;

}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-164d5e3903a81cb4d838e9589ae8cfe9024804362067e9acdcd574526a4f4b7a>

Завдання №7 **Self-Practice Work**

Деталі по програмі + Вставка з кодом з підписами до вставки. Посилання на файл програми у пул-запиті GitHub

#include <iostream>

*void* multiplyMatrices(*int*\*\* *A*, *int*\*\* *B*, *int*\*\* *C*, *int* *n*, *int* *m*, *int* *q*) {

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

for (*int* j = 0; j < *q*; ++j) {

*C*[i][j] = 0;

for (*int* k = 0; k < *m*; ++k) {

*C*[i][j] += *A*[i][k] \* *B*[k][j];

}

}

}

}

*void* printMatrix(*int*\*\* *matrix*, *int* *rows*, *int* *cols*) {

for (*int* i = 0; i < *rows*; ++i) {

for (*int* j = 0; j < *cols*; ++j) {

std::cout << *matrix*[i][j] << " ";

}

std::cout << std::endl;

}

}

*int* main() {

using *namespace* std;

*int* n, m, q;

cout << "Введіть розмірність матриці A (n m): ";

cin >> n >> m;

cout << "Введіть розмірність матриці B (m q): ";

cin >> m >> q;

*int*\*\* A = new *int*\*[n];

*int*\*\* B = new *int*\*[m];

*int*\*\* C = new *int*\*[n];

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

A[i] = new *int*[m];

C[i] = new *int*[q];

for (*int* j = 0; j < m; ++j) {

cout << "A[" << i << "][" << j << "]: ";

cin >> A[i][j];

}

}

for (*int* i = 0; i < m; ++i) {

B[i] = new *int*[q];

for (*int* j = 0; j < q; ++j) {

cout << "B[" << i << "][" << j << "]: ";

cin >> B[i][j];

}

}

multiplyMatrices(A, B, C, n, m, q);

cout << "Матриця A:" << endl;

printMatrix(A, n, m);

cout << "Матриця B:" << endl;

printMatrix(B, m, q);

cout << "Результат множення матриць A та B:" << endl;

printMatrix(C, n, q);

// Звільнення пам'яті

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

delete[] A[i];

delete[] C[i];

}

delete[] A;

delete[] B;

delete[] C;

return 0;

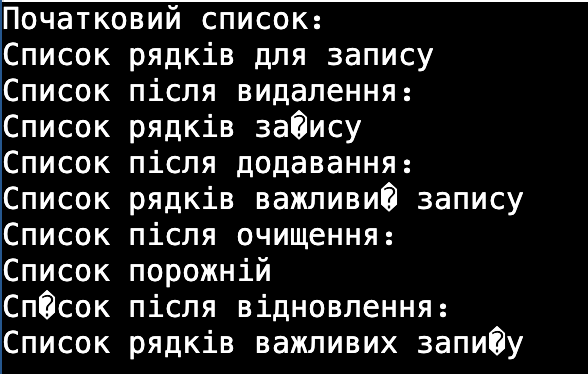
}

Підпис та № до блоку з кодом програми

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1186/files#diff-6098299a4f0763ce7de1e56799820df6c359cb014dba8ca7c770ecb51058c722>

**5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 **VNS Lab 10 - Task 1-11** Деталі по виконанню і тестуванню програми



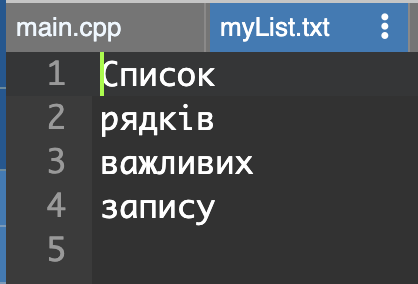


Рисунок 9. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **2 години**

Завдання №2 **Algotester Lab 5** Деталі по виконанню і тестуванню програми

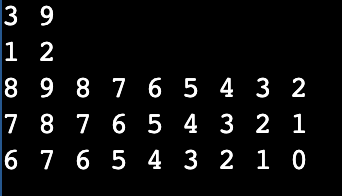


Рисунок 10. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **2 години**

Завдання №3 **Algotester Lab 7-8** Деталі по виконанню і тестуванню програми

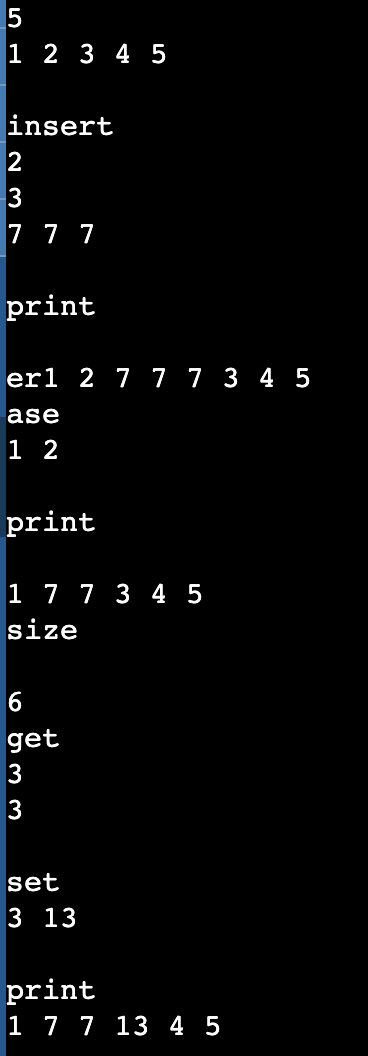


Рисунок 11. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **2 години**

Завдання №4 **Class Practice Work** Деталі по виконанню і тестуванню програми

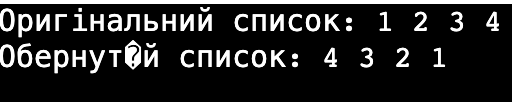


Рисунок 12. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **2 години**

Завдання №5 **Class Practice Wor** Деталі по виконанню і тестуванню програми



Рисунок 13. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **2 години**

Завдання №6 **Class Practice Work** Деталі по виконанню і тестуванню програми

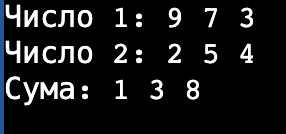


Рисунок 14. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **2 години**

Завдання №7 **Self-Practice Work** Деталі по виконанню і тестуванню програми

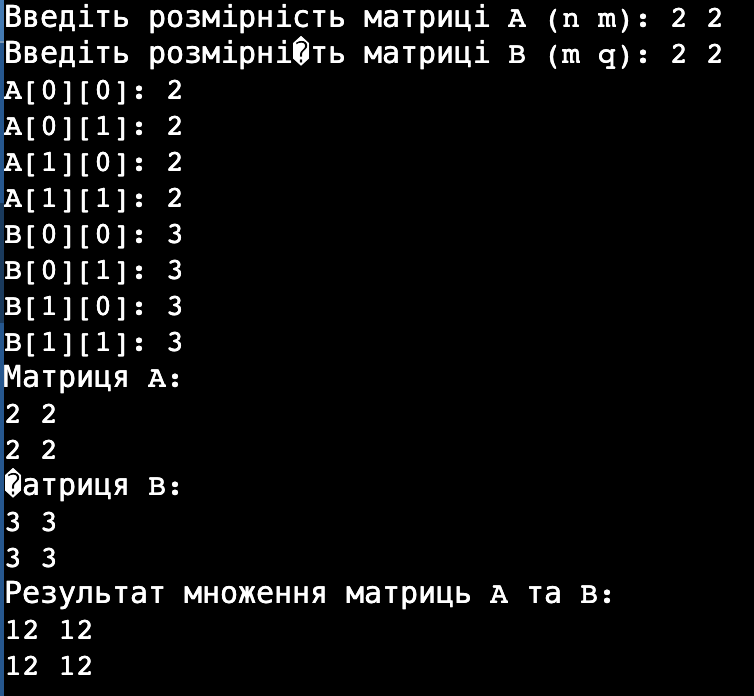
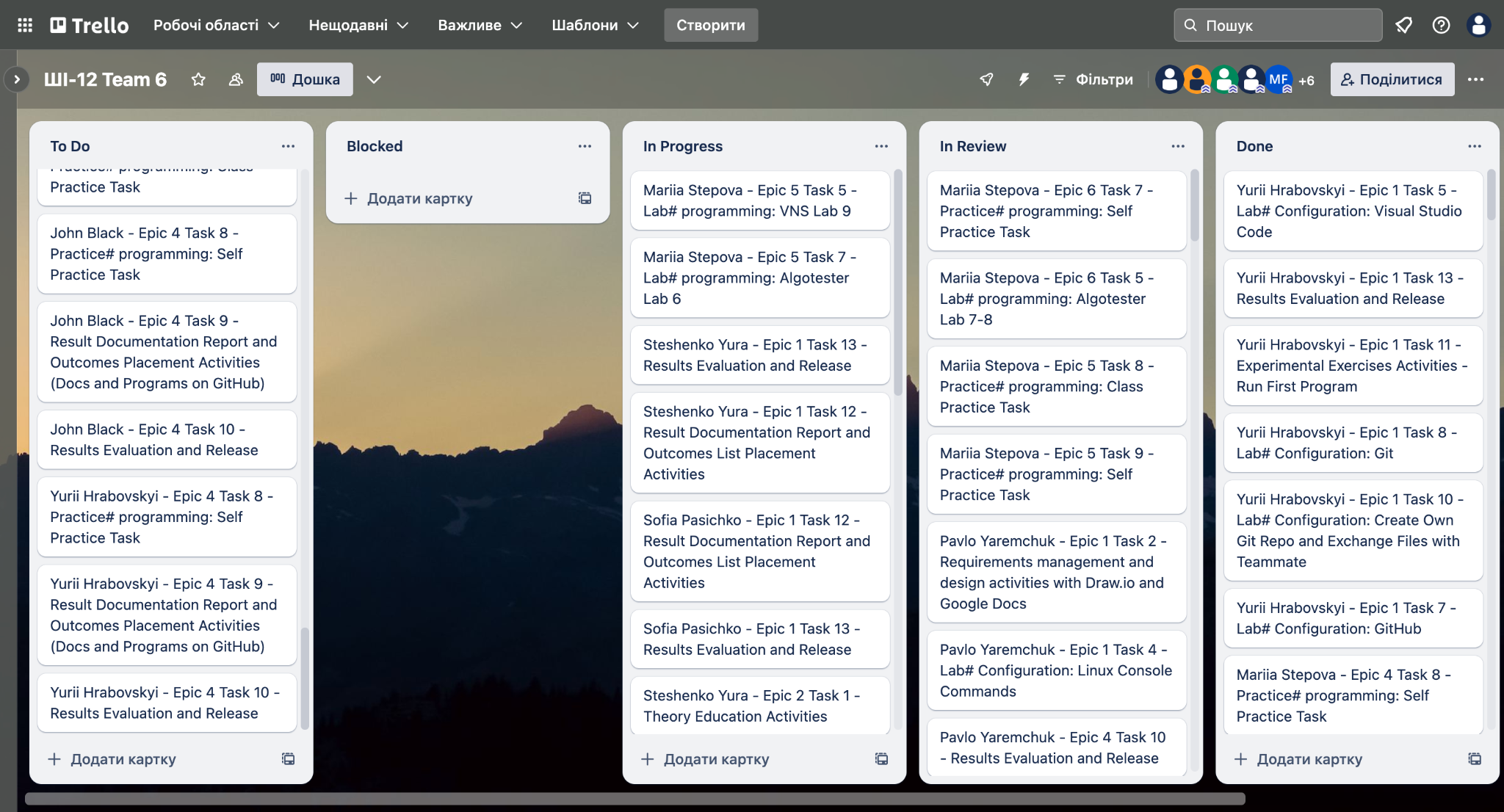
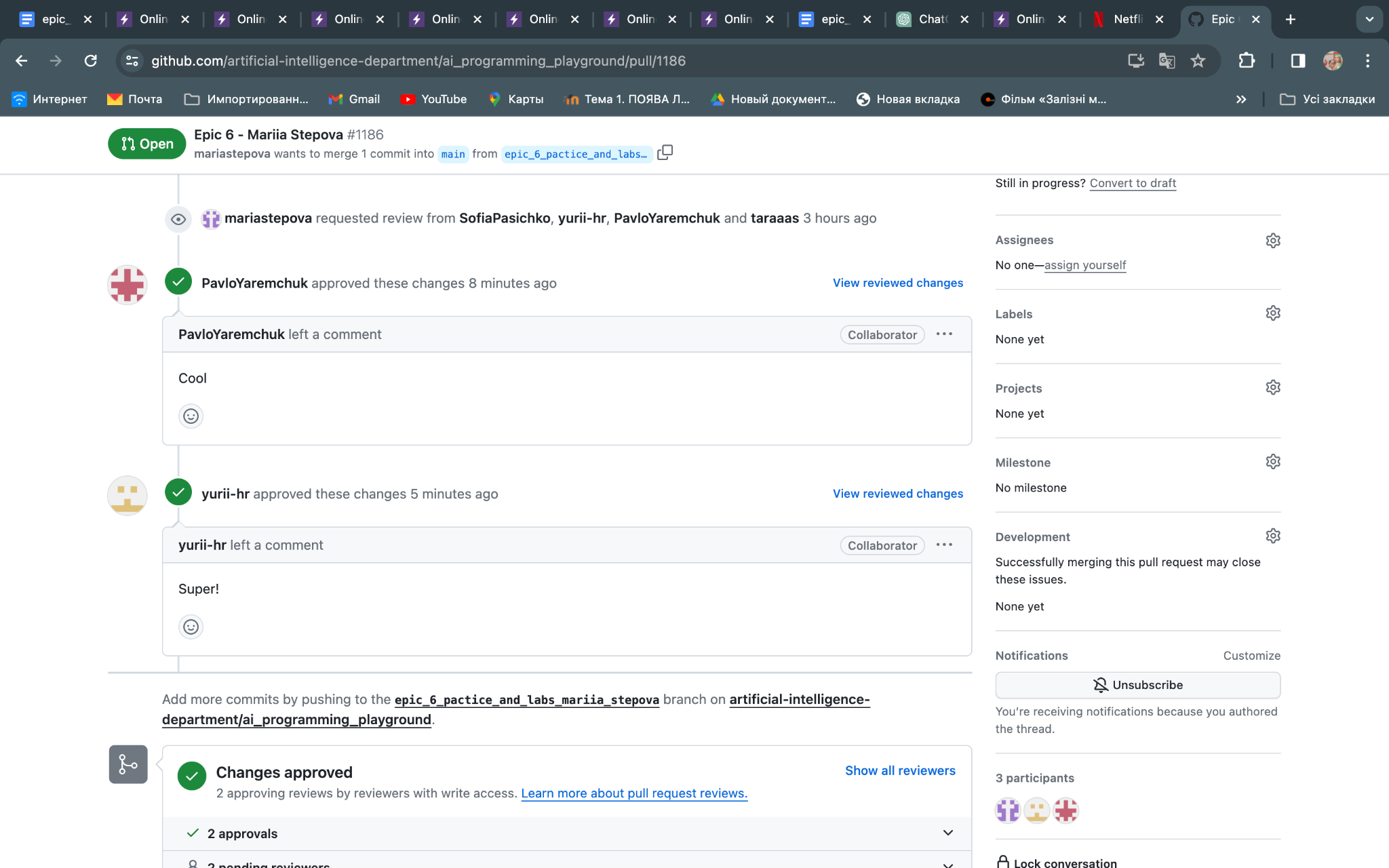


Рисунок 15. Скриншот результату виконання завдання

Час затрачений на виконання завдання: **3 години**

## **6. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* 
* 
* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи



# **Висновки:**

Розуміння динамічних структур даних, таких як списки, дерева - дозволяє оптимізувати використання пам'яті та виконання складних операцій над даними. Засвоєні алгоритми сортування, пошуку та обходу даних допомогли мені виконати лабораторні та практичні роботи та вибирати оптимальні методи для їх розв'язання.