Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

**про виконання лабораторних та практичних робіт блоку № 6**

На тему:  «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

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

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт № 6

**Виконав:**

Студент групи ШІ-13

Ласкавий Артем Олександрович

# **Тема роботи:**

Динамічні структури у програмуванні. Стек. Черга. Зв’язаний двонаправлений та однонаправлений списки. Бінарне дерево. Бінарне дерево пошуку.

# **Мета роботи:**

Зрозуміти принципи роботи із динамічними структурами даних. Детальніше дізнатися про поняття двонаправленого і однонапрaвленого списку, та операцій роботи з ними . Ознайомитися із теорією щодо бінарних дерев, та особливостей будови бінарного дерева пошуку. Ознайомитися з чергою та стеком. Застосувати на практиці отримані знання.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічний масив.
* Тема №2: Черга. Стек.
* Тема №3: Однонаправлений список. Двонаправлений список.
* Тема №4: Бінарні дерева. Бінарні дерева пошуку.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічний масив.
  + Джерела Інформації
    - <https://acode.com.ua/urok-90-dynamichni-masyvy/>
    - Л.І. Мочурад Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина 1»
  + Що опрацьовано:
    - Поняття динамічного масиву, методи його задання. Виділення пам’яті для динамічного масиву, вивільнення пам’яті. Витік пам’яті.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.2023
  + Звершення опрацювання теми: 15.12.2023
* Тема №2: Однонаправлений список. Двонаправлений список.
  + Джерела Інформації:
    - Л.І. Мочурад Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина 1»
    - <https://erudyt.net/navchalni-predmety/informatika/prohramuvannya/linijnyj-odnozvyaznyj-spysok.html>
    - <https://www.bestprog.net/uk/2022/02/16/c-linear-doubly-linked-bidirectional-list-general-concepts-ua/>
  + Що опрацьовано:
    - Поняття однозв’язного та двозвязного списків. Структура вузла для обох типів. Операції над списками: вставка, виділення, пошук, перебір.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 21.11.2023
  + Звершення опрацювання теми: 15.12.2023
* Тема №3: Бінарні дерева. Бінарні дерева пошуку.
  + Джерела Інформації:
    - <https://www.programiz.com/dsa/binary-tree>
    - <https://uk.myservername.com/binary-search-tree-c>
    - <https://www.youtube.com/watch?v=Gt2yBZAhsGM>
  + Що опрацьовано:
    - Поняття дерев та їх структури. Бінарне дерево, особливості його вузлів. Бінарне дерево пошуку та його властивості.
  + Статус: Ознайомлений
  + Початок опрацювання теми:12.12.2023
  + Звершення опрацювання теми: 15.12.2023

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант 17

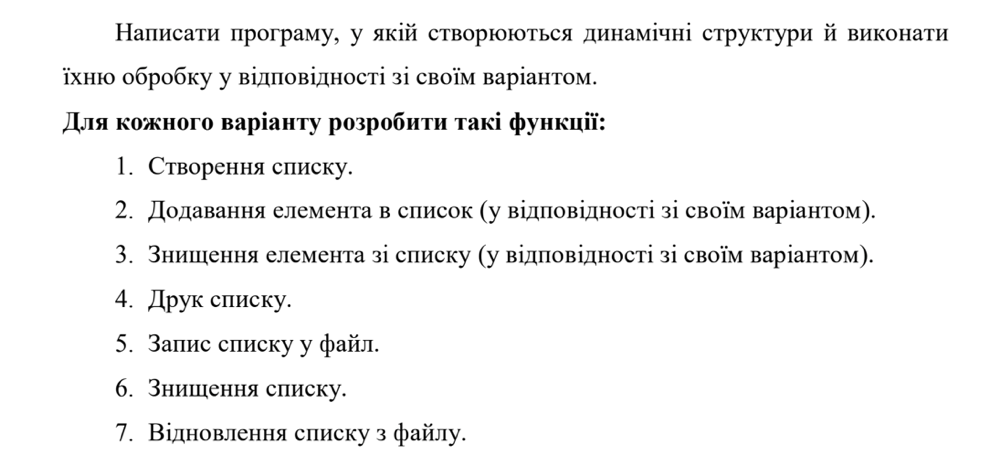


Рисунок - Умова Завдання №1 VNS Lab 10

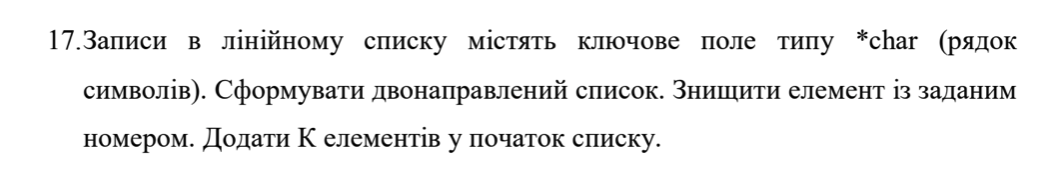


Рисунок - Умова Завдання №1 VNS Lab 10

Завдання №2 Algotester Lab 5v2

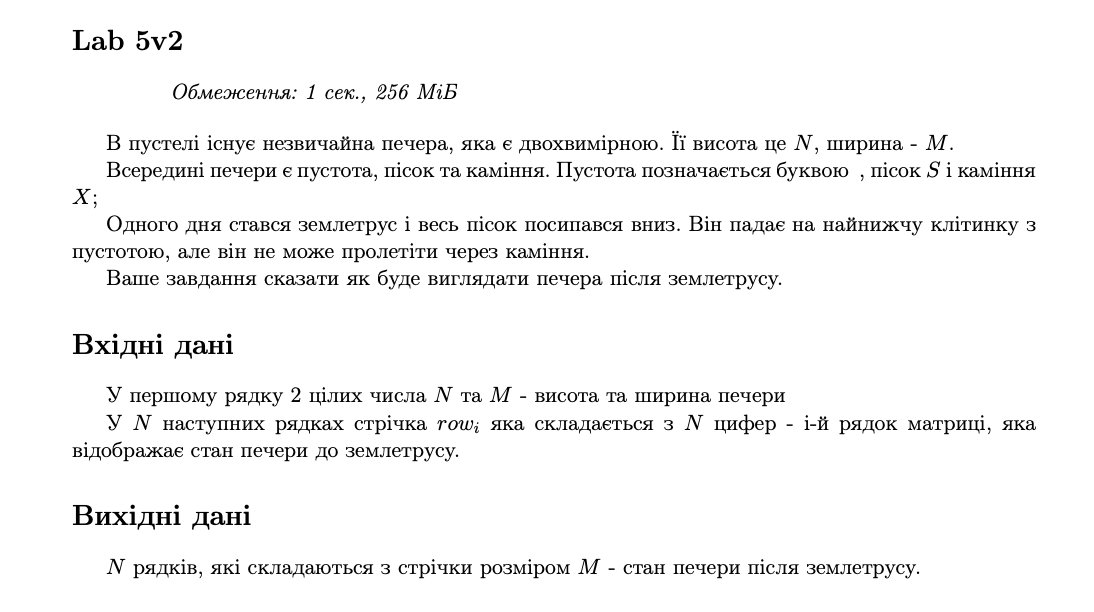
* Варіант 2
* Деталі завдання
* 

Рисунок - Умова Завдання №2 Algotester Lab 5v2

Завдання №3 Algotester Lab 78

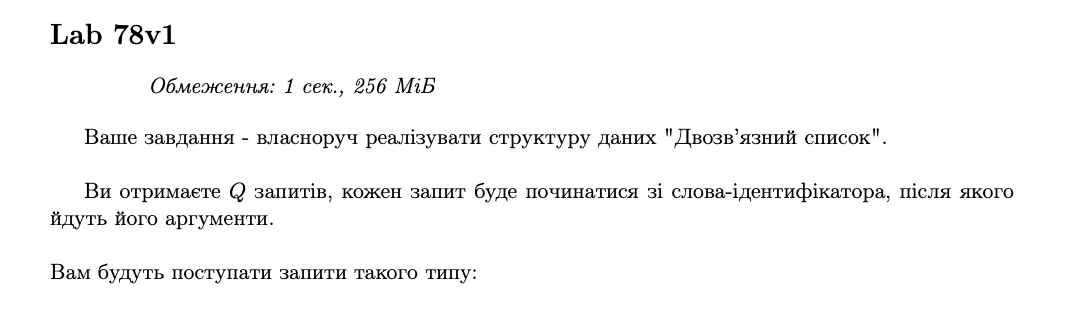
* Варіант 1
* Деталі завдання
* 

Рисунок Умова Завдання №2 Algotester Lab 78v1

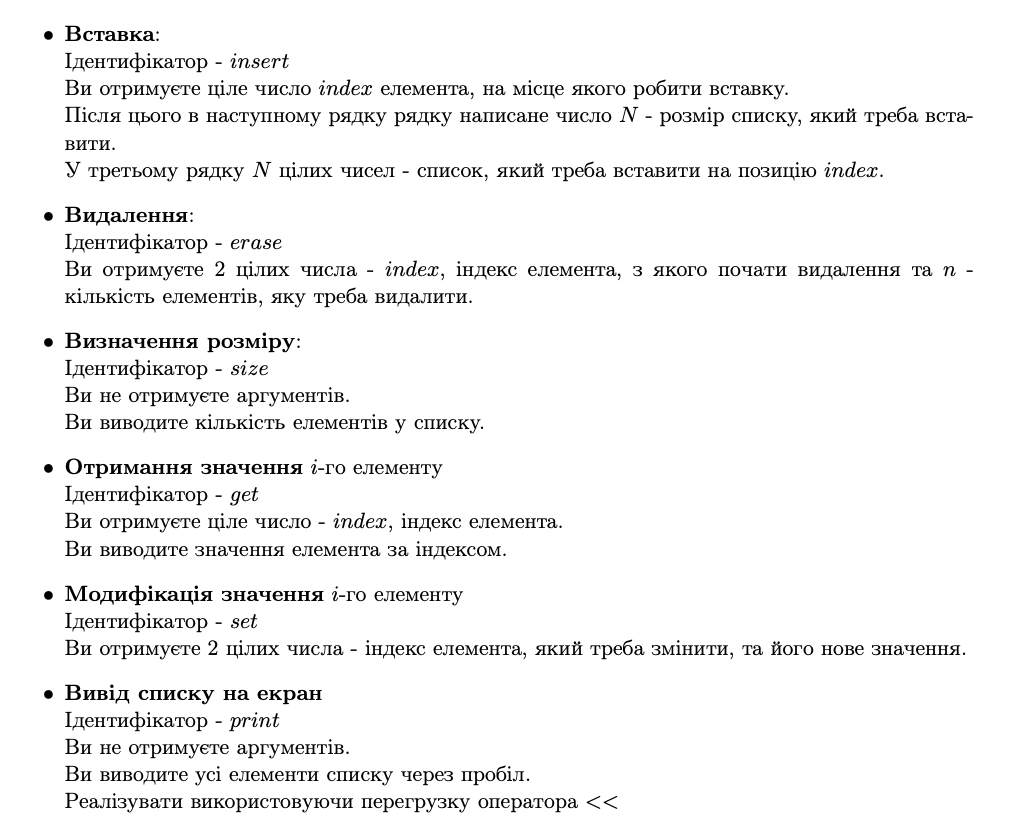
* 

Рисунок Умова Завдання №2 Algotester Lab 78v1

Завдання №4 Class Practice Tasks 1-3

Задача №1 - Реверс списку (Reverse list)

***Реалізувати метод реверсу списку:*** Node\* reverse(Node \*head);

*Умови задачі:*

-       використовувати цілочисельні значення в списку;

-       реалізувати метод реверсу;

* -       реалізувати допоміжний метод виведення вхідного і обернутого списків;

Задача №2 - Порівняння списків

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

*Умови задачі:*

-       використовувати цілочисельні значення в списку;

-       реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;

-       якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

## Задача №3 – Додавання великих чисел

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

*Умови задачі:*

-       використовувати цифри від 0 до 9 для значень у списку;

-       реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379  ⟹  9→7→3);

-       функція повертає новий список, передані в функцію списки не модифікуються.

Завдання №5 Class Practice Tasks 4-5

* Деталі завдання :

## Задача №4 - Віддзеркалення дерева

TreeNode \*create\_mirror\_flip(TreeNode \*root);

*Умови задачі:*

-       використовувати цілі числа для значень у вузлах дерева

-       реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева

-       функція повертає нове дерево, передане в функцію дерево не модифікується

## Задача №54 - Записати кожному батьківському вузлу суму підвузлів

void tree\_sum(TreeNode \*root);

*Умови задачі:*

-       використовувати цілочисельні значення у вузлах дерева;

-       реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів

-       вузол-листок не змінює значення

-       значення змінюються від листків до кореня дерева

Завдання №5 Algotester Self practice Cвічки на Halloween 0607

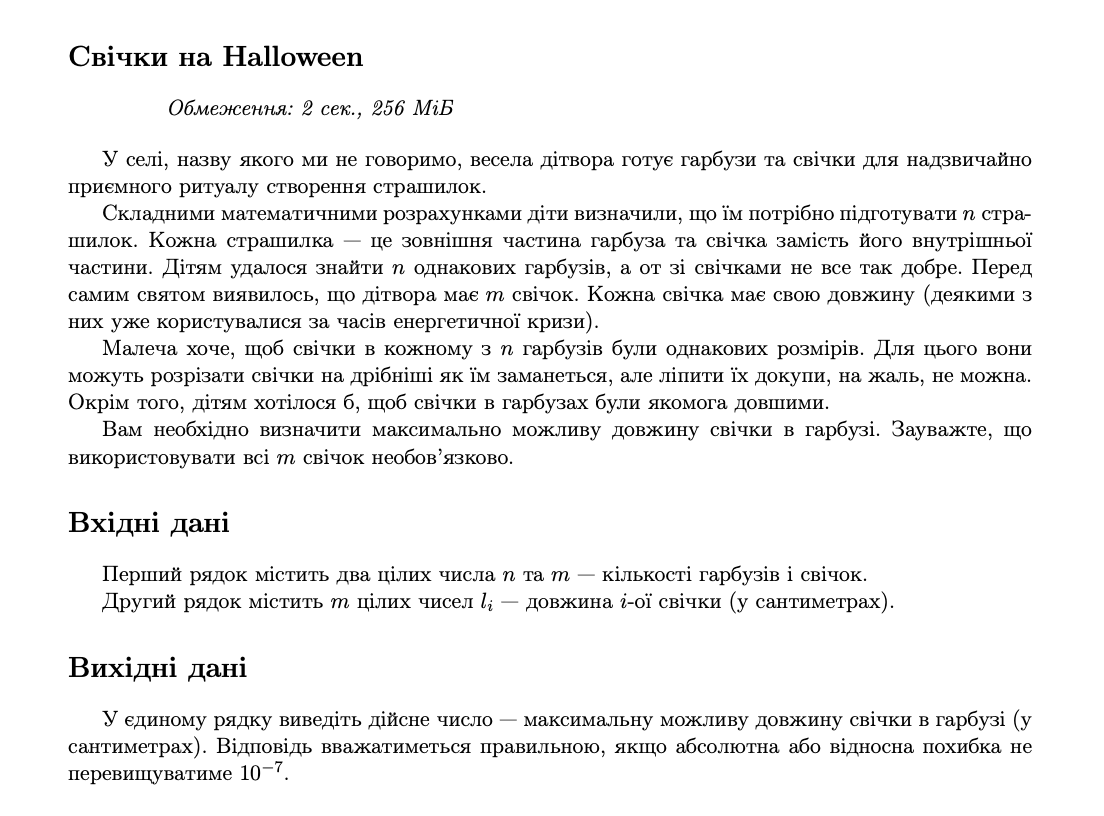


Рисунок - Умова Завдання №5 Algotester self practice

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

Програма №1 VNS Lab 10

* Планований час на реалізацію: 3 години

Програма №2 Algotester Lab 5v2

* Блок-схема:

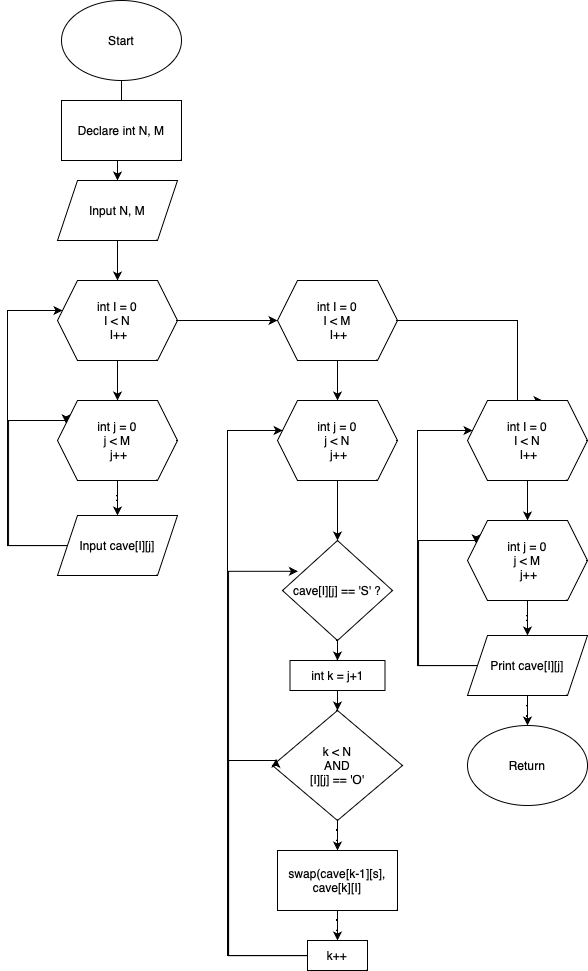


Рисунок - Блок-схема завдання №2 Algotester Lab 5v2

* Планований час на реалізацію: 2 години

Програма №3 Algotester Lab 78

* Планований час на реалізацію: 4 години

Програма №4 Class Practice Tasks 1-3

* Планований час на реалізацію: 2 години

Програма №5 Class Practice Tasks 4-5

* Планований час на реалізацію: 2 години

Програма №6 Algotester Self practice Cвічки на Halloween 0607

* Планований час на реалізацію: 2 години

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

Завдання №1 VNS Lab 10

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_artem_laskavyi/ai_13/artem_laskavyi/Epic%206/vns_lab_10_task_1_variant_1_john_black.cpp>

#include <iostream>

#include <cstring>

#include <fstream>

#include <string>

using namespace std;

struct Node {

char\* data;

Node\* next;

Node\* prev;

Node(const char\* str) : next(nullptr), prev(nullptr) {

data = new char[strlen(str)];

strcpy(data, str);

}

~Node() {

delete[] data;

}

};

class DoublyLinkedList {

private:

Node\* head;

public:

DoublyLinkedList() : head(nullptr) {}

void createList() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

cin.ignore();

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

char\* data = new char[256];

cin.getline(data, 256);

addToEnd(data);

}

}

void printList() {

if (!head) {

cout << "List is empty." << endl;

return;

}

Node\* current = head;

while (current) {

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

current = current->next;

}

cout << endl;

}

void addToEnd(const char\* data) {

Node\* newNode = new Node(data);

if (!head) {

head = newNode;

} else {

Node\* current = head;

while (current->next) {

current = current->next;

}

current->next = newNode;

newNode->prev = current;

}

}

void deleteNodeAtPosition(int position) {

if (!head) {

cout << "List is already empty." << endl;

return;

}

if (position == 1) {

Node\* temp = head;

head = head->next;

if (head) {

head->prev = nullptr;

}

delete temp;

} else {

int count = 1;

Node\* current = head;

while (count < position && current) {

current = current->next;

++count;

}

if (!current) {

cout << "Invalid position." << endl;

return;

}

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

if (current->next) {

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

}

delete current;

}

}

void saveToFile(const char\* fileName) {

ofstream file(fileName);

if (file.is\_open()) {

Node\* current = head;

while (current) {

file << current->data << "\n";

current = current->next;

}

file.close();

cout << "List saved to file successfully." << endl;

} else {

cout << "Unable to open file." << endl;

}

}

void clearList() {

Node\* current = head;

while (current) {

Node\* temp = current;

current = current->next;

delete temp;

}

head = nullptr;

cout << "List deleted successfully." << endl;

}

void restoreFromFile(const char\* fileName) {

ifstream file(fileName);

if (file.is\_open()) {

string data;

while (getline(file, data)) {

addToEnd(data.c\_str());

}

file.close();

cout << "List restored from file successfully." << endl;

} else {

cout << "Unable to open file." << endl;

}

}

};

int main() {

DoublyLinkedList myList;

myList.createList();

cout << "List after creation: ";

myList.printList();

int positionToDelete;

cout << "Enter position to delete: ";

cin >> positionToDelete;

myList.deleteNodeAtPosition(positionToDelete);

cout << "List after deletion of node: ";

myList.printList();

myList.saveToFile("list.txt");

myList.clearList();

cout << "List after deletion: ";

myList.printList();

myList.restoreFromFile("list.txt");

cout << "List after restoration: ";

myList.printList();

return 0;

}

Рисунок : Код до програми №1

Завдання №2 Algotester Lab 5

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_artem_laskavyi/ai_13/artem_laskavyi/Epic%206/algotester_lab_5_variant_1_artem_laskavyi.cpp>

#include <iostream>

using namespace std;

int main() {

int N, M; // N - rows, M - columns

cin >> N >> M;

char cave[N][M];

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

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

cin >> cave[i][j];

}

}

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

for (int j = N - 1; j >= 0; j--) {

if (cave[j][i] == 'S') {

int k = j + 1;

while (k < N && cave[k][i] == 'O') {

swap(cave[k - 1][i], cave[k][i]);

k++;

}

}

}

}

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

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

cout << cave[i][j];

}

cout << endl;

}

return 0;

}

Рисунок : Код до програми №2

Завдання №3 Algotester Lab 78

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_artem_laskavyi/ai_13/artem_laskavyi/Epic%206/algotester_lab_78_variant_1_artem_laskavyi.cpp>

#include <iostream>

#include <string>

using namespace std;

class Node{

public:

int data;

Node\* next;

Node\* prev;

};

class DoublyLinkedList {

private:

public:

Node\* head;

DoublyLinkedList() : head(nullptr) {}

// Function to create a new node with given data

Node\* createNode(int data){

Node\* new\_node = new Node();

new\_node->data = data;

new\_node->next = nullptr;

new\_node->prev = nullptr;

return new\_node;

}

// Function to find a node at a specific index

Node\* getNode(Node\* head, int index){

int count = 0;

Node\* current = head;

while (current != nullptr && count < index)

{

current = current->next;

count++;

}

return current;

}

void insertAfter(Node\* prev\_node, int new\_data)

{

if (prev\_node == nullptr){

Node\* new\_node = createNode(new\_data);

new\_node->next = head;

new\_node->prev = nullptr;

if (head != nullptr)

head->prev = new\_node;

head = new\_node;

}else{

Node\* new\_node = createNode(new\_data);

new\_node->next = prev\_node->next;

prev\_node->next = new\_node;

new\_node->prev = prev\_node;

if (new\_node->next != nullptr)

new\_node->next->prev = new\_node;

}

}

void deleteNode(Node\*\* head\_ref, Node\* del)

{

if (\*head\_ref == nullptr || del == nullptr)

return;

if (del->prev != nullptr)

del->prev->next = del->next;

else

\*head\_ref = del->next;

if (del->next != nullptr)

del->next->prev = del->prev;

delete del;

}

void setNode(Node\* head, int index, int new\_data)

{

Node\* node = getNode(head, index);

node->data = new\_data;

}

int getSize(Node\* head)

{

int count = 0;

Node\* current = head;

while (current != nullptr)

{

count++;

current = current->next;

}

return count;

}

int getData(Node\* head, int index){

Node\* node = getNode(head, index);

return node->data;

}

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

cout << endl;

}

};

int main()

{

int q;

cin >> q;

DoublyLinkedList list;

for (int i = 1; i <= q; i++)

{

string input;

cin >> input;

cin.ignore();

if (input == "insert")

{

int index, length, element;

cin >> index >> length;

for (int j = 0; j < length; j++)

{

cin >> element;

Node\* prev\_node = list.getNode(list.head, index - 1);

list.insertAfter(prev\_node, element);

index++;

}

}

else if (input == "erase")

{

int eraseIndex, eraseLength;

cin >> eraseIndex >> eraseLength;

Node\* startNode = list.getNode(list.head, eraseIndex);

Node\* endNode = list.getNode(list.head, eraseIndex + eraseLength);

while (startNode != nullptr && startNode != endNode)

{

Node\* temp = startNode->next;

list.deleteNode(&list.head, startNode);

startNode = temp;

}

}

else if (input == "size")

{

cout << list.getSize(list.head) << endl;

}

else if (input == "get")

{

int getIndex;

cin >> getIndex;

cout << list.getData(list.head, getIndex) << endl;

}

else if (input == "set")

{

int setIndex, setData;

cin >> setIndex >> setData;

list.setNode(list.head, setIndex, setData);

}

else if (input == "print")

{

list.printList(list.head);

}

}

return 0;

}

Рисунок : Код до програми №3

Завдання №4 Class Practice Tasks 1-3

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_artem_laskavyi/ai_13/artem_laskavyi/Epic%206/practice_work_task_1_artem_laskavyi.cpp>

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

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

};

class LinkedList {

public:

Node\* head;

// Конструктор за замовчуванням

LinkedList() : head(nullptr) {}

// Метод для додавання елемента в кінець списку

void addNode(int val) {

Node\* newNode = new Node(val);

if (!head) {

head = newNode;

} else {

Node\* current = head;

while (current->next) {

current = current->next;

}

current->next = newNode;

}

}

void reverse() {

Node\* prev = nullptr;

Node\* current = head;

Node\* next = nullptr;

while (current) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

void printList() {

Node\* current = head;

while (current) {

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

current = current->next;

}

cout << endl;

}

bool compare(Node\* h2) {

Node\* h1 = head;

while (h1 && h2) {

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

return false; // Якщо дані в вузлах не співпадають

}

h1 = h1->next;

h2 = h2->next;

}

// Якщо один список закінчується раніше іншого, повертаємо false

if (h1 || h2) {

return false;

}

return true; // Всі елементи співпали, повертаємо true

}

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

LinkedList result;

int rem = 0;

while (n1 || n2 || rem) {

int sum = (n1 ? n1->data : 0) + (n2 ? n2->data : 0) + rem;

rem = sum / 10;

result.addNode(sum % 10);

if (n1) n1 = n1->next;

if (n2) n2 = n2->next;

}

return result.head;

}

};

int main() {

LinkedList myList1;

myList1.addNode(1);

myList1.addNode(2);

myList1.addNode(3);

myList1.addNode(4);

LinkedList myList2;

myList2.addNode(4);

myList2.addNode(3);

myList2.addNode(2);

myList2.addNode(1);

cout << "Original List 1: ";

myList1.printList();

myList1.reverse();

cout << "Reversed List 1: ";

myList1.printList();

if (myList1.compare(myList2.head)) {

cout << "Lists are equal." << endl;

}else {

cout << "Lists are not equal." << endl;

}

LinkedList result1;

result1.head = result1.nodeSum(myList1.head, myList2.head);

cout << "Sum: ";

result1.printList();

return 0;

}

Рисунок : Код до програми №4

Завдання №5 Class Practice Tasks 4-5

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_artem_laskavyi/ai_13/artem_laskavyi/Epic%206/practice_work_task_2_artem_laskavyi.cpp>

#include <iostream>

using namespace std;

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val) : data(val), left(nullptr), right(nullptr) {}

};

TreeNode\* create\_mirror\_flip(TreeNode\* root){

// Base case: If the tree is empty, return nullptr

if (!root) {

return nullptr;

}

// Recursive case: Swap the left and right subtrees

TreeNode\* temp = root->left;

root->left = create\_mirror\_flip(root->right);

root->right = create\_mirror\_flip(temp);

return root;

}

// Helper function to print the tree (in-order traversal)

void print\_tree(TreeNode\* root){

if (root) {

print\_tree(root->left);

cout << root->data << " "; //printinf current(parent) node

print\_tree(root->right);

}

}

void delete\_tree(TreeNode\* root){

if (root) {

delete\_tree(root->left);

delete\_tree(root->right);

delete root;

}

}

void tree\_sum(TreeNode\* root){

if (!root || (!root->left && !root->right)) {

// Якщо вузол порожній або є листком, не змінюємо його значення

return;

}

// Рекурсивно викликаємо для лівого та правого піддерева

tree\_sum(root->left);

tree\_sum(root->right);

// Змінюємо значення поточного батьківського вузла на суму значень підвузлів

root->data = (root->left ? root->left->data : 0) + (root->right ? root->right->data : 0);

}

int main(){

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

root->left->left = new TreeNode(4);

root->left->right = new TreeNode(5);

root->right->left = new TreeNode(6);

root->right->right = new TreeNode(7);

cout << "Original Tree: ";

print\_tree(root);

cout << endl;

TreeNode\* mirroredTree = create\_mirror\_flip(root);

cout << "Mirrored Tree: ";

print\_tree(mirroredTree);

cout << endl;

tree\_sum(root);

cout << "Tree after Sum Calculation: ";

print\_tree(root);

cout << endl;

delete\_tree(root);

return 0;

}

Рисунок : Код до програми №5

Завдання №6 Algotester Self practice Cвічки на Halloween 0607

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_artem_laskavyi/ai_13/artem_laskavyi/Epic%206/practice_work_self_algotester_tasks_artem_laskavyi.cpp>

#include <iostream>

#include <stdio.h>

#include <algorithm>

#include <cmath>

#include <vector>

using namespace std;

int main() {

int n, m, candle; // m - pumpkins, n - candles

cin >> m >> n;

vector<int> a;

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

cin >> candle;

a.push\_back(candle);

}

long double low = 0, high = \*max\_element(a.begin(), a.end()), mid = 0, mid\_prev = 0;

int count = 0;

long double epsilon = 0.00000001;

mid = (high + low) / 2;

do{

mid\_prev = mid;

count = 0;

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

if (a[i] > mid) {

count += floor(a[i]/mid);

}

}

if (count < m) {

high = mid;

}else{

low = mid;

}

mid = (high + low) / 2;

}while (abs(mid\_prev - mid) > epsilon);

printf("%.10Lf", mid);

return 0;

}

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

Завдання №1 VNS Lab 10

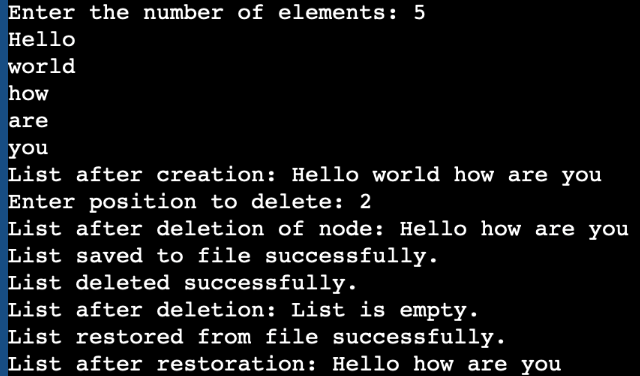


Рисунок : Тестування програми №1

Час затрачений на виконання завдання: 1 день

Завдання №2 Algotester Lab 5

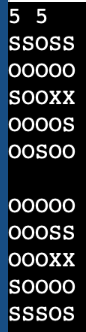


Рисунок : Тестування програми №2

Час затрачений на виконання завдання: 1 день

Завдання №3 Algotester Lab 78

Рисунок : Тестування програми №3

Час затрачений на виконання завдання: 1 день

Завдання №4 Class Practice Tasks 1-3

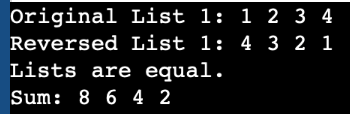


Рисунок : Тестування програми №4

Час затрачений на виконання завдання: 1 день

Завдання №5 Class Practice Tasks 4-5

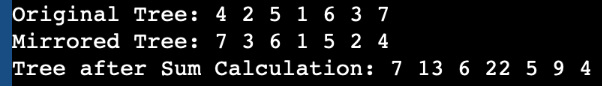


Рисунок : Тестування програми №5

Час затрачений на виконання завдання: 1 день

Завдання №6 Algotester Self practice Cвічки на Halloween 0607

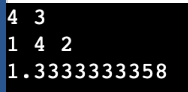
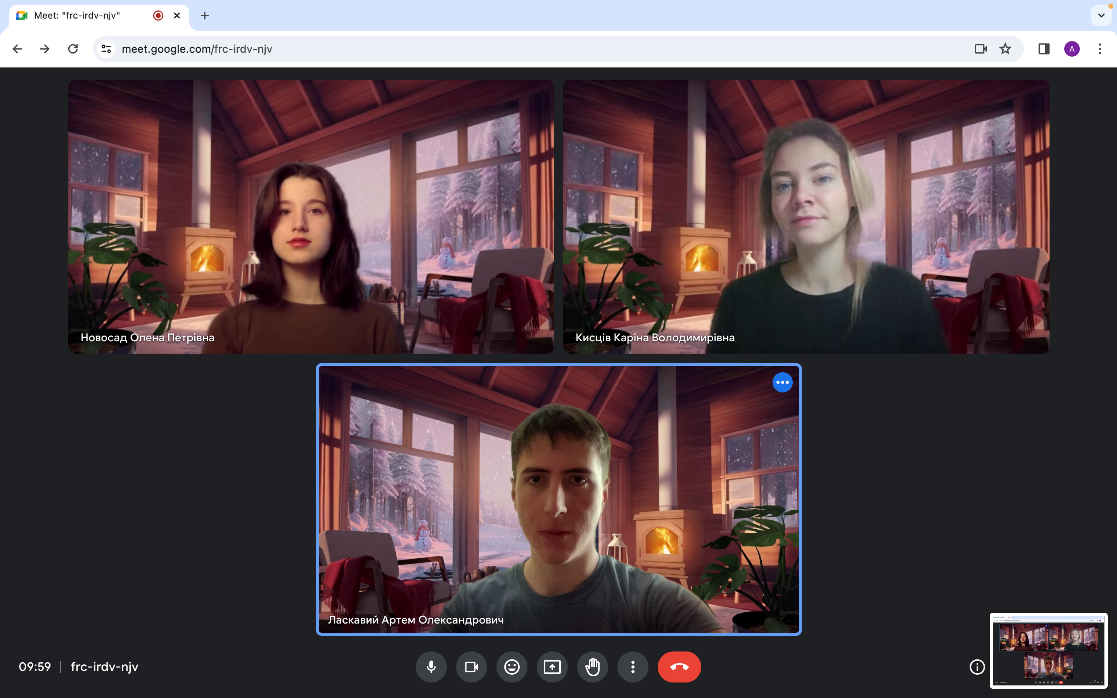
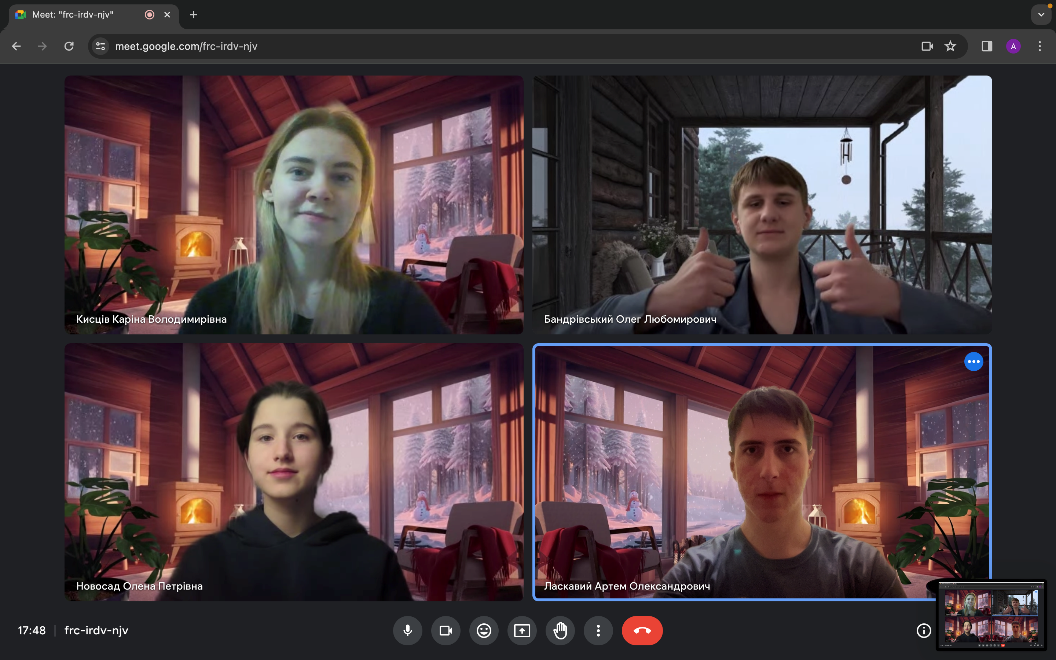
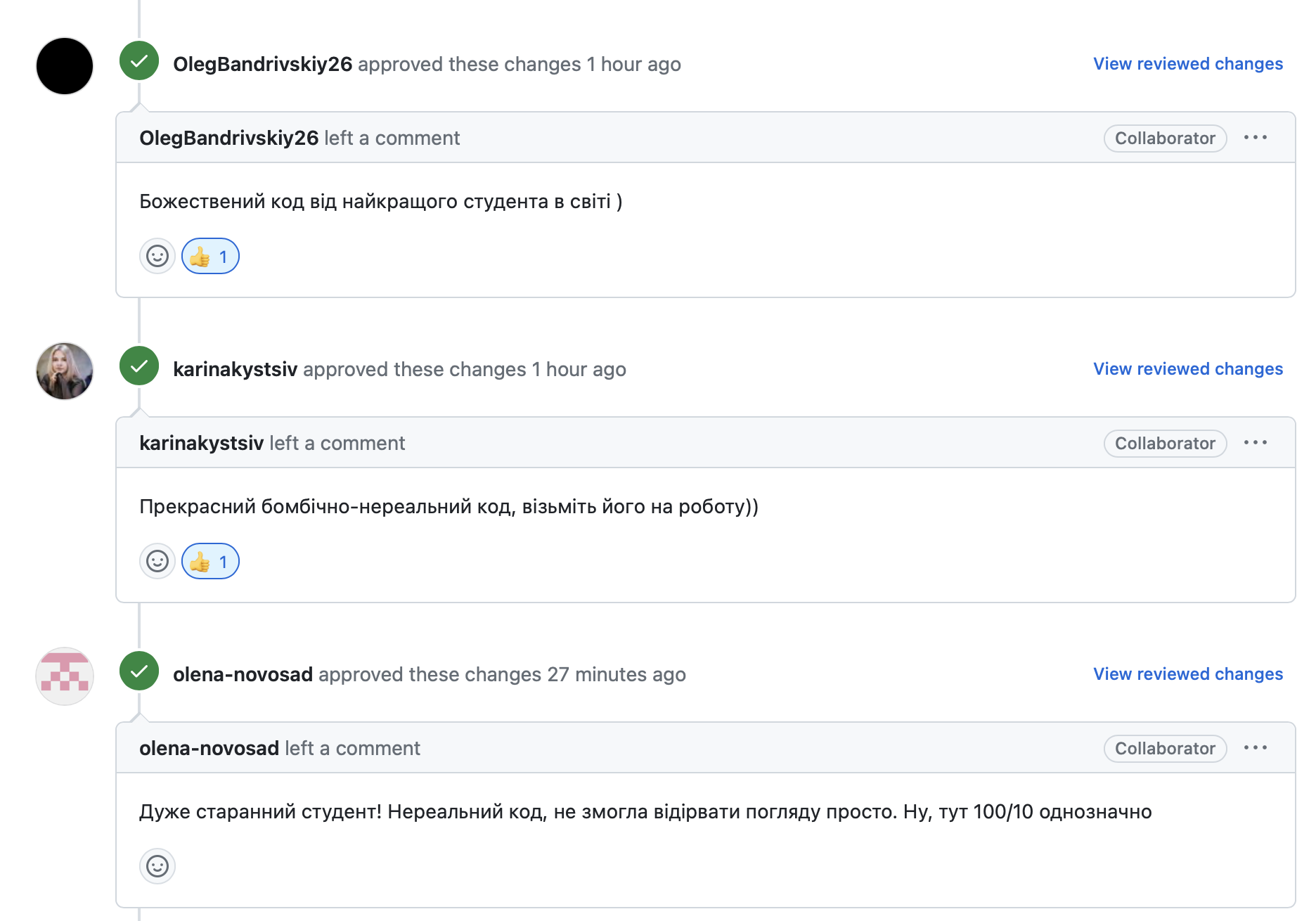
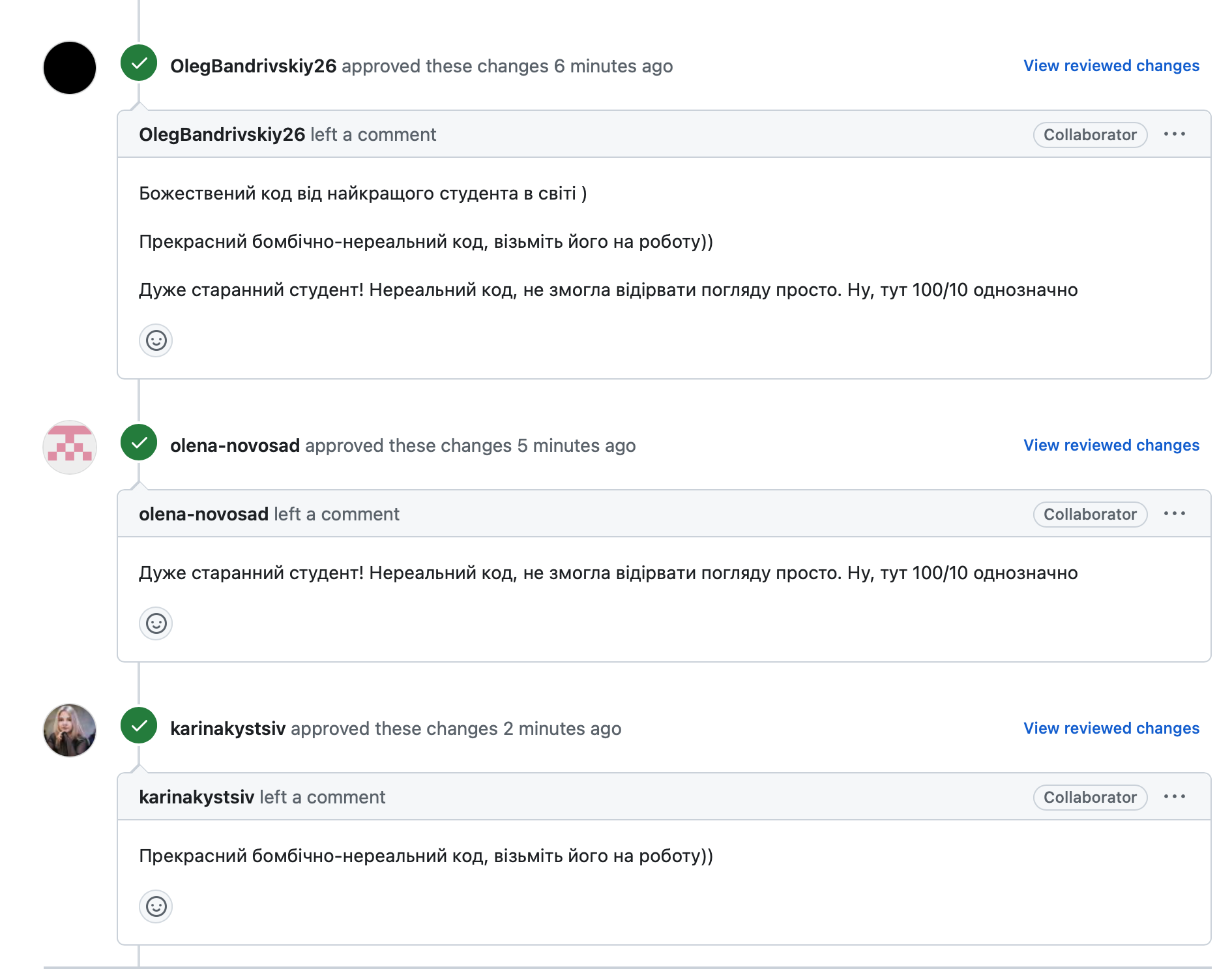


Рисунок - Тестування програми №6

## **5. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло



* Скрін з 2-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:**

Під час виконання даної лабораторної роботи я детальніше дізнався про методи роботи із однонаправленими та двонаправленими списками. Ознайомився із поняттям бінарного дерева і застосувавце на практиці.