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

***Виконала:***

студентка групи ШІ-12

Лебединська Яна Олександрівна

# ***Тема роботи:***

Огляд основних принципів динамічних структур даних. Переваги використання списків, бінарних дерев та стеків у порівнянні з іншими структурами. Реалізація однонаправлених та двонаправлених списків. Використання стеку для реалізації алгоритмів та структурних рішень. Визначення оптимальних випадків для використання кожної структури.

# ***Мета роботи:***

Дослідження та систематизація інформації щодо динамічних структур даних, зокрема списків, бінарних дерев та стеків, та розгляд їхніх алгоритмів обробки. Розкриття основних принципів реалізації цих структур, вивчення їхніх переваг та недоліків, а також дослідження алгоритмів обробки даних структур, зокрема в контексті бінарних дерев.

# ***Теоретичні відомості:***

1. Теоретичні відомості з переліком важливих тем:

* Тема №1:Однозв’язні списки
* Тема №2: Двозв’язні списки
* Тема №3: Бінарні дерева
* Тема №4: Стек

1. Індивідуальний план опрацювання теорії:

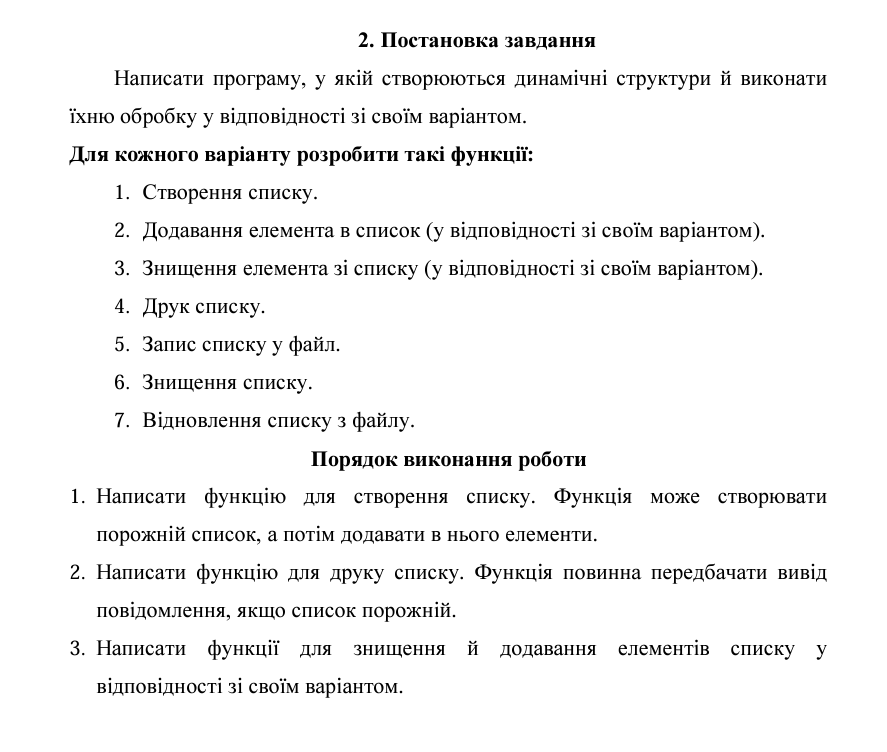
* Тема №1:Однозв’язні списки
  + Джерела Інформації
    - [C++ Теорія ⦁ Урок 139 ⦁ ADT ⦁ Однозв'язний список](https://www.youtube.com/watch?v=-25REjF_atI)
  + Що опрацьовано:
    - Що таке однозв’язний список
    - Особливості роботи з ним
    - Реалізація структури вузла однозв’язного списку
  + Статус: Ознайомлена
  + Початок опрацювання теми: 09.12.23
  + Звершення опрацювання теми: 10.12.23
  + по темі зроблено конспект
* Тема №2:Двозв’язні списки
  + Джерела Інформації:
    - [C++ Теорія ⦁ Урок 140 ⦁ ADT ⦁ Двозв'язний список](https://www.youtube.com/watch?v=QLzu2-_QFoE&t=1971s)
  + Що опрацьовано:
    - Що таке двозв’язний список
    - Особливості роботи з ним
    - Реалізація структури вузла двозв’язного списку
  + Статус: Ознайомлена
  + Початок опрацювання теми: 10.12.23
  + Звершення опрацювання теми: 10.12.23
  + По темі зроблено конспект
* Тема №3: Бінарні дерева
  + Джерела Інформації:
    - [C++ Теорія ⦁ Урок 144 ⦁ ADT ⦁ Бінарне дерево](https://www.youtube.com/watch?v=qBFzNW0ALxQ&t=1495s)
  + Що опрацьовано:
    - що таке бінарне дерево
    - особливості роботи з бінарними деревами
    - реалізація структури вузла бінарного дерева
  + Статус: Ознайомлена
  + Початок опрацювання теми: 10.12.23
  + Звершення опрацювання теми: 11.12.23
  + по темі зроблено конспект
* Тема №4: Стек
  + Джерела Інформації:
    - <https://acode.com.ua/urok-111-stek-i-kupa/> (**стаття**)
    - <https://www.youtube.com/watch?v=ZYvYISxaNL0> (**відео**)
  + Що опрацьовано:
    - що таке стек
    - особливості стеку
    - робота зі стеком
  + Статус: Ознайомлена
  + Початок опрацювання теми: 11.12.23
  + Звершення опрацювання теми: 11.12.23
  + По темі зроблено конспект.
* Тема №5: Класи. Шаблони класів
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=CUGTegwAwz4> (**відео**)
    - <https://acode.com.ua/urok-121-klasy-ob-yekty-i-metody/> (**стаття**)
    - <https://acode.com.ua/urok-183-shablony-klasiv/> (**стаття**)
  + Що опрацьовано:
    - Що таке класи та шаблони класів
    - що таке конструктор
    - чим зручніші шаблони класів
    - реалізації класів та шаблонів класів
  + Статус: Ознайомлена
  + Початок опрацювання теми: 09.12.23
  + Звершення опрацювання теми: 09.12.23

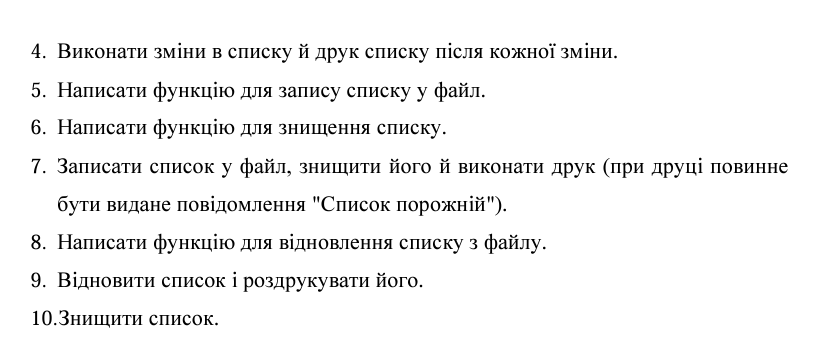
# ***Виконання роботи:***

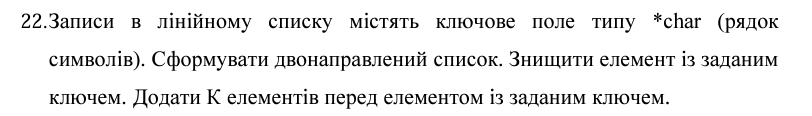
## **1. Опрацювання завдання та вимог до програм та середовища:**

*Завдання №1*: VNS Lab 10

* Варіант №22
* постановка завдання:



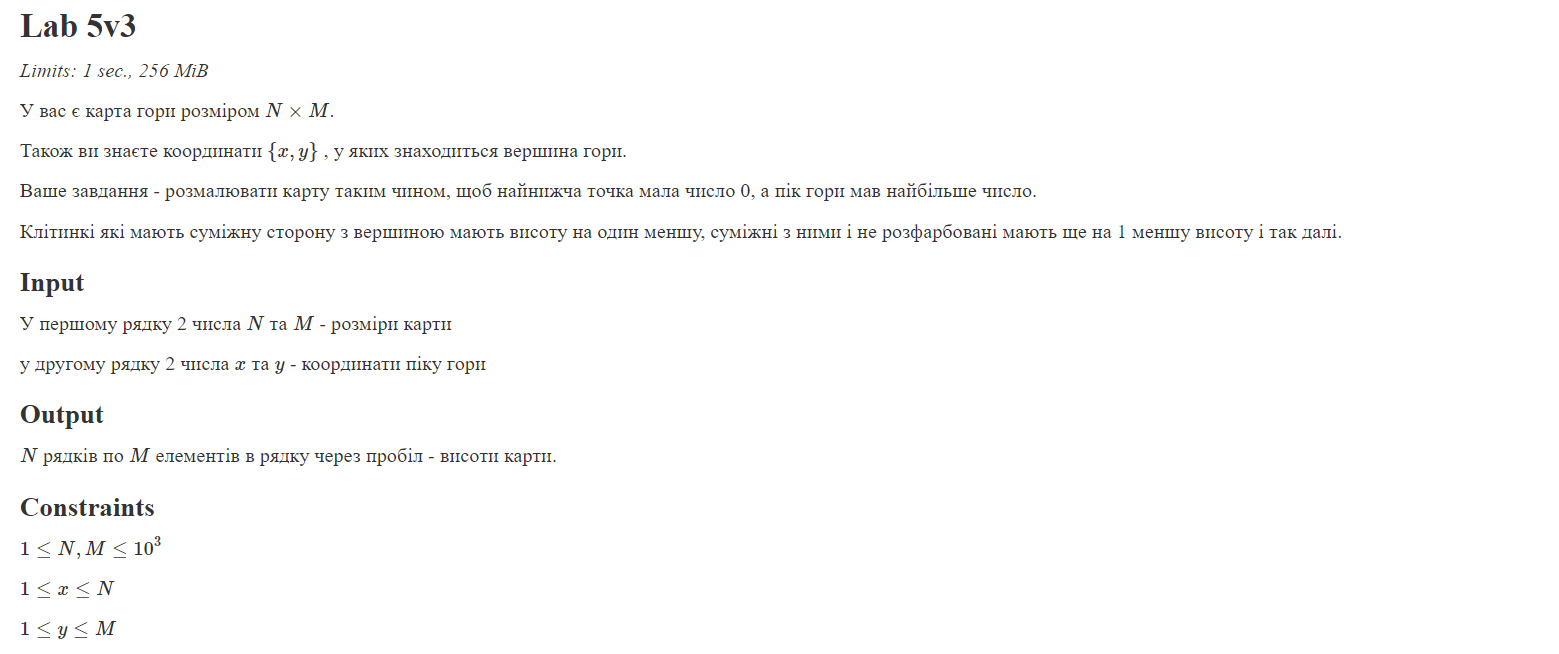




*умова завдання lab 10 v22*

*Завдання №2*: Algotester lab 5

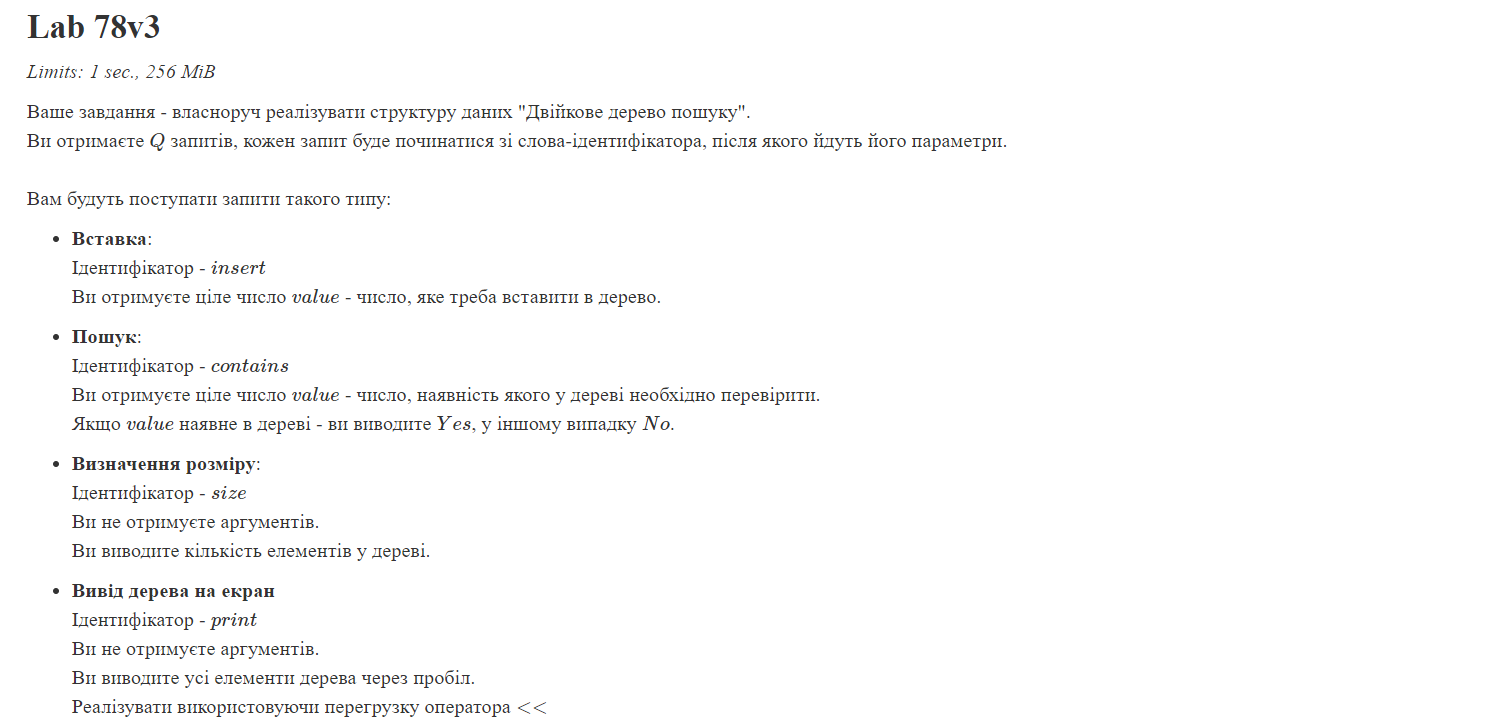
* Варіант №3
* постановка завдання: <https://algotester.com/en/ContestProblem/DisplayWithEditor/134644>



*умова завдання lab 5 v3 algotester*

*Завдання №3*: Algotester lab 7\_8

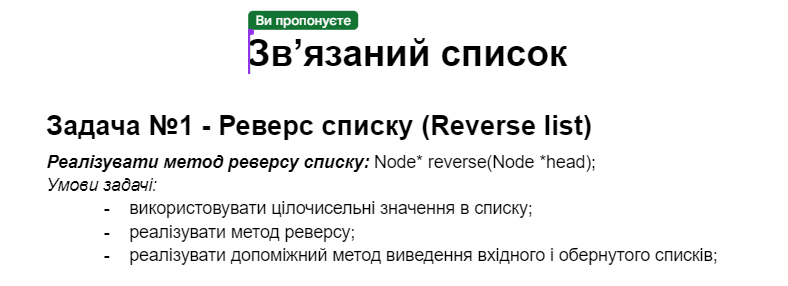
* Варіант №3
* постановка завдання: <https://algotester.com/en/ContestProblem/DisplayWithEditor/134649>

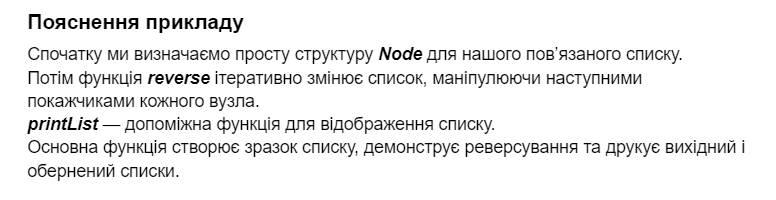


*умова завдання lab 7\_8 v3 algotester*

*Завдання №4*:Class Practice Task 1

* без варіанту
* постановка завдання:

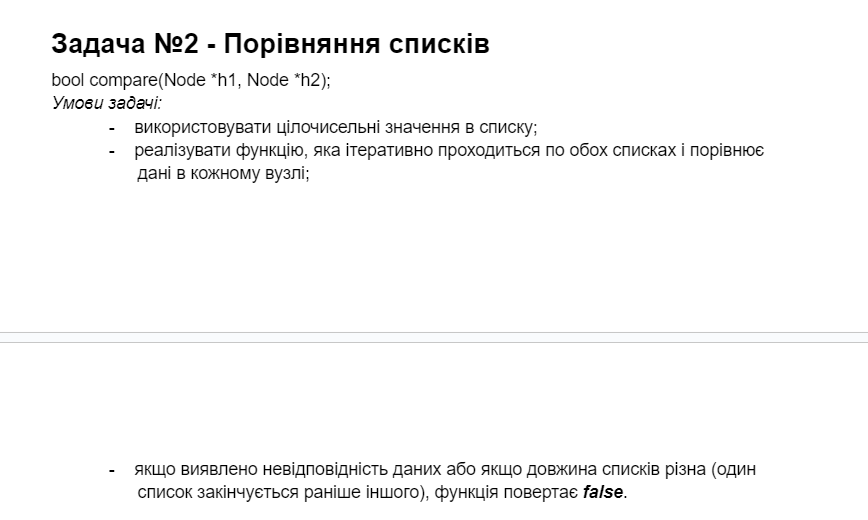


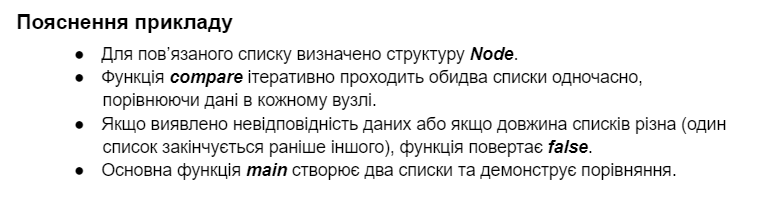


*умова задачі class practice 1*

*Завдання №5*: Class Practice Task 2

* без варіанту
* постановка завдання:

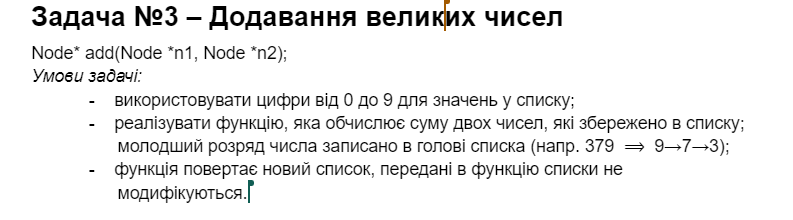




*умова задачі class practice 2*

*Завдання №6*: Class Practice Task 3

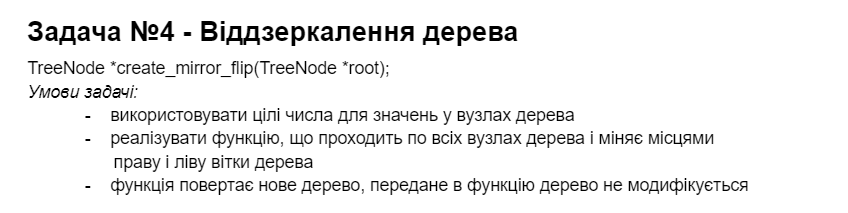
* Без варіанту
* постановка завдання:



*умова задачі class practice 3*

*Завдання №7*: Class Practice Task 4

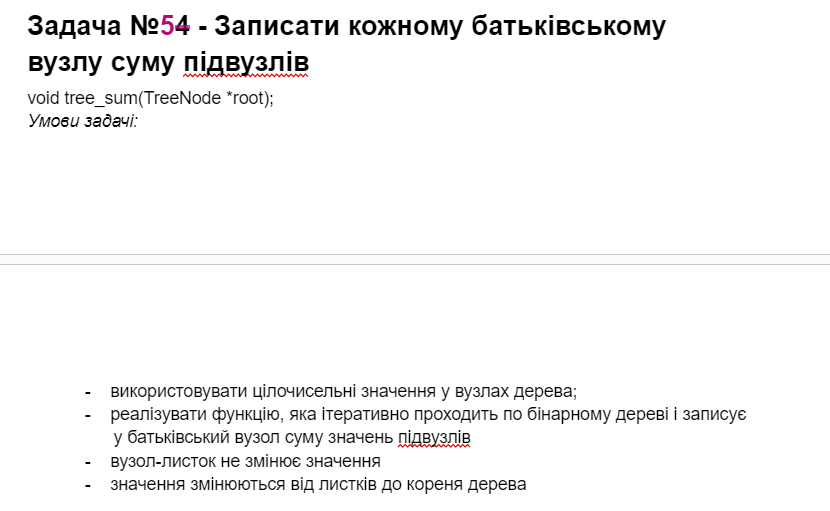
* Без варіанту
* постановка завдання:



*умова задачі class practice 4*

*Завдання №8*: Class Practice Task 5

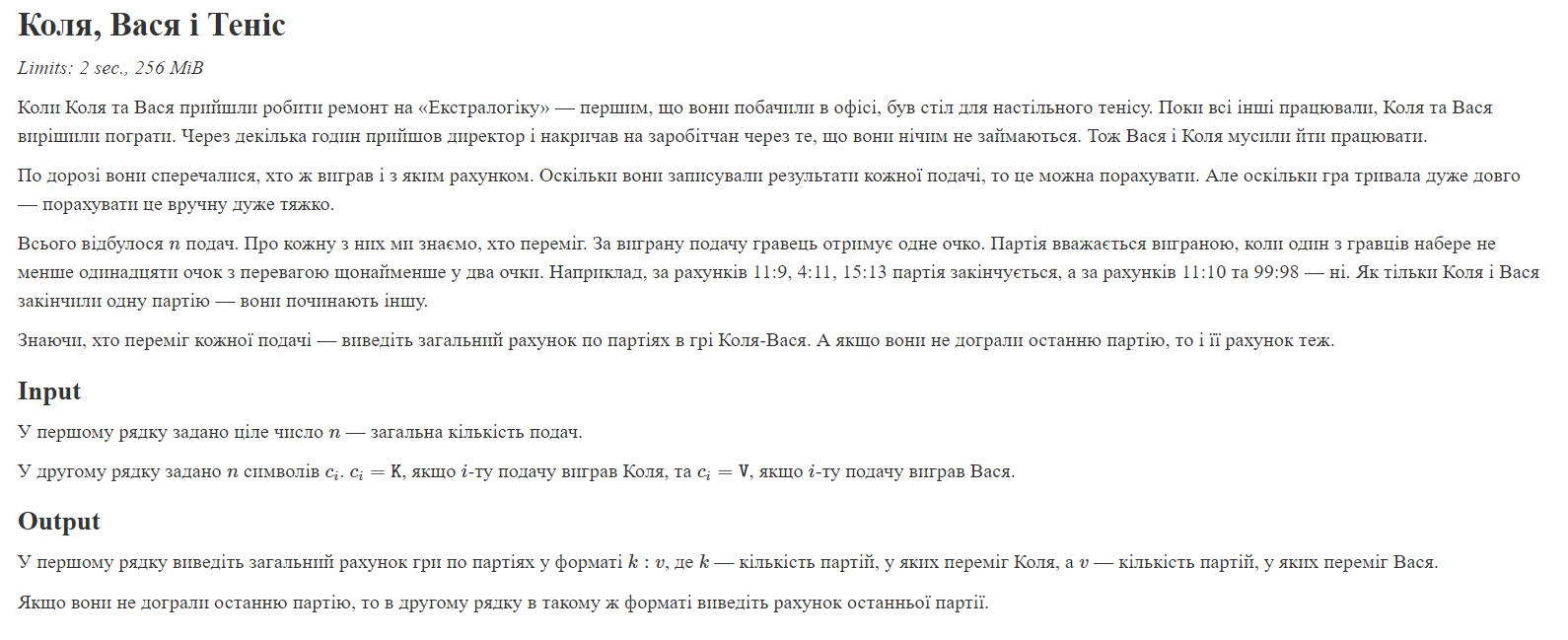
* Без варіанту
* постановка завдання:



*умова задачі class practice 5*

*Завдання №*9 Self Practice Task (algotester)

* Без варіанту
* Постановка завдання: <https://algotester.com/en/ArchiveProblem/DisplayWithEditor/13>



*умова задачі self practice Algotester*

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

***Програма №1*** VNS Lab 10

* Планований час на реалізацію: 1 год 20 хв

***Програма №2*** Algotester lab 5

* Планований час на реалізацію 1 год

***Програма №3***Algotester lab 7\_8

* Планований час на реалізацію 1 год 20 хв

***Програма №4*** Class Practice Task 1

* Планований час на реалізацію: 40 хв

***Програма №5*** Class Practice Task 2

* Планований час на реалізацію: 40 хв

***Програма №6*** Class Practice Task 3

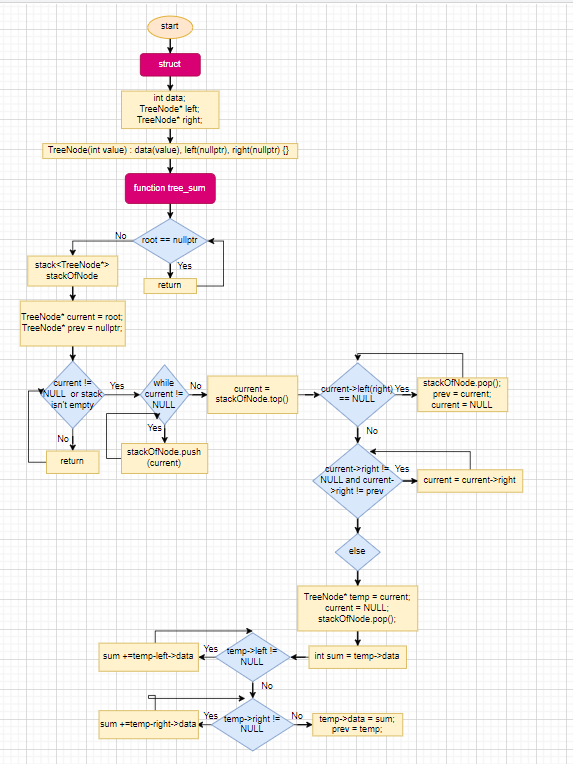
* Планований час на реалізацію: 40 хв

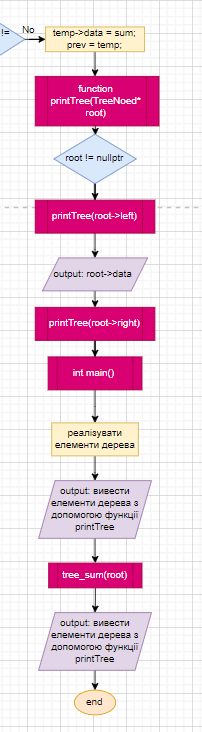
***Програма №6*** Class Practice Task 4

* Планований час на реалізацію: 40 хв

***Програма №6*** Class Practice Task 5

* Блок-схема





* Планований час на реалізацію: 50 хв

***Програма №7*** Self Practice Task (algotester)

Планований час на реалізацію: 30 хв

## **3. Конфігурація середовища до виконання завдань:**

Додаткової конфігурації не потрібно.

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

**Завдання №1** VNS Lab 10

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/vns_lab_10_v22_yana_lebedynska.cpp>

#include <iostream>

#include <fstream>

#include <cstring>

#include <sstream>

#include <string>

using namespace std;

struct Node

{

char\* data;

Node\* previous;

Node\* next;

};

//створення пустого списоку

Node\* createNewList()

{

return nullptr;

}

//додавання елеемнту в кінець списку

void add(Node\* &head, const char\* element)

{

Node\* newNode = new Node;

newNode->data = new char[strlen(element) + 1];

strcpy(newNode->data, element); //копіює рядок element в newNode->data

newNode->next = nullptr;

if (head == nullptr)

{

head = newNode;

newNode->previous = nullptr;

}

else

{

Node\* temporary = head;

while (temporary->next != nullptr)

{

temporary = temporary->next;

}

temporary->next = newNode;

newNode->previous = temporary;

}

}

//вивести на екран весь список

void print(Node\* head)

{

if (head == nullptr)

{

cout << "The list is empty" << endl;

return;

}

Node\* current = head;

while (current != nullptr)

{

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

current = current->next;

}

cout << endl;

}

//видалення елемента

void deleteElement(Node\* &head, const char\* key)

{

Node\* current = head;

while (current != nullptr && strcmp(current->data, key) != 0)

{

current = current->next;

}

if (current == nullptr)

{

cout << "Element with the key " << key << " isn't found!" << endl;

return;

}

if (current->previous != nullptr)

{

current->previous->next = current->next;

}

else

{

head = current->next;

}

if (current->next != nullptr)

{

current->next->previous = current->previous;

}

delete[] current->data; //видалення виділеної пам'яті

delete current; //видалення самого елемента

}

//додавання елемента перед вказаним елементом

void addKElemBeforeKey(Node\* &head, const char\* key, int k, const char\* elements)

{

Node\* current = head;

while (current != nullptr && strcmp(current->data, key) != 0)

{

current = current->next;

}

if (current == nullptr)

{

cout << "Element with the key " << key << " isn't found!" << endl;

return;

}

istringstream iss(elements);

string element;

for (int i = 0; i < k && iss >> element; ++i)

{

Node\* newNode = new Node;

newNode->data = new char[element.length() + 1];

strcpy(newNode->data, element.c\_str());

if (current->previous != nullptr)

{

current->previous->next = newNode;

}

else

{

head = newNode;

}

newNode->previous = current->previous;

newNode->next = current;

if (current->previous != nullptr)

{

current->previous = newNode;

}

}

}

//запис списку у файл

void writeToFile(Node\* head, const char\* nameOfFile)

{

ofstream file\_output(nameOfFile);

if (!file\_output.is\_open())

{

cout << "File couldn't be opened!" << endl;

return;

}

Node\* current = head;

while (current != nullptr)

{

file\_output << current->data << endl;

current = current->next;

}

cout << "List is written to file '" << nameOfFile << "'" << endl;

file\_output.close();

}

//видалення цілого списку

void deleteList(Node\* &head)

{

while (head != nullptr)

{

Node\* temporary = head;

head = head->next;

delete[] temporary->data;

delete temporary;

}

}

//перезапис списку з файлу

Node\* restoreList(const char\* nameOfFile)

{

ifstream file\_input(nameOfFile);

Node\* head = nullptr;

if (!file\_input.is\_open())

{

cout << "File couldn't be opened!" << endl;

return nullptr;

}

string line;

while (getline(file\_input, line))

{

add(head, line.c\_str());

}

file\_input.close();

cout << "List is restored from file '" << nameOfFile << "'" << endl;

return head;

}

int main()

{

Node\* List = createNewList();

int operation;

do

{

cout << endl;

cout << "Choose the operation (enter a number): " << endl;

cout << "1. Add element" << endl;

cout << "2. Delete element" << endl;

cout << "3. Add element before another element" << endl;

cout << "4. Print whole list" << endl;

cout << "5. Write list to file" << endl;

cout << "6. Restore list from file" << endl;

cout << "7. Delete the entire list" << endl;

cout << "0. Exit" << endl;

cout << "Enter the operation number -> ";

cin >> operation;

cin.ignore();

switch (operation)

{

case 1:

{

cout << endl;

cout << "Enter the element you want to add: ";

string element;

getline(cin, element);

add(List, element.c\_str());

break;

}

case 2:

{

cout << endl;

cout << "Enter the key of the element you want to delete: ";

string key;

getline(cin, key);

deleteElement(List, key.c\_str());

break;

}

case 3:

{

cout << endl;

cout << "Enter the key before which you want to add elements: ";

string key;

getline(cin, key);

int k;

cout << "Enter the number of elements you want to add: ";

cin >> k;

cin.ignore();

cout << "Enter the elements you want to add: ";

string ELEMENTS;

getline(cin, ELEMENTS);

addKElemBeforeKey(List, key.c\_str(), k, ELEMENTS.c\_str());

break;

}

case 4:

{

cout << endl;

cout << "Your current list: ";

print(List);

break;

}

case 5:

{

cout << endl;

cout << "Enter the name of the file in which you want to write the list: ";

string fileName;

getline(cin, fileName);

writeToFile(List, fileName.c\_str());

break;

}

case 6:

{

cout << endl;

cout << "Enter the name of the file from which you want to restore the list: ";

string fileName;

getline(cin, fileName);

deleteList(List); //очистити існуючий список

List = restoreList(fileName.c\_str());

print(List);

break;

}

case 7:

{

cout << endl;

deleteList(List);

List = createNewList();

cout << "The entire list now is deleted!" << endl;

break;

}

case 0:

{

cout << "The end of the program!" << endl;

break;

}

default:

{

cout << endl;

cout << "Operation not found! Try again." << endl;

break;

}

}

}

while (operation != 0);

deleteList(List);

return 0;

}

*код задачі lab 10*

**Завдання №2** Algotester lab 5 v3

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/algotester_lab_5_yana_lebedynska.cpp>

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

int main()

{

int N, M, x, y;

cin >> N >> M;

cin >> x >> y;

vector<vector<int>> peak(N, vector<int> (M, 0));

int heaightOfPeak = 0;

for(int X = 0; X < N; X++)

{

for(int Y = 0; Y < M; Y++)

{

peak[X][Y] = abs(x - 1 - X) + abs(y - 1 - Y);

heaightOfPeak = max(heaightOfPeak, peak[X][Y]);

}

}

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

{

for(int l = 0; l < M; l++)

{

peak[i][l] = heaightOfPeak - peak[i][l];

}

}

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

{

for(int l = 0; l < M; l++)

{

cout << peak[i][l] << " ";

}

cout << endl;

}

return 0;

}

}

## *код задачі algotester lab 5 v3*

**Завдання №3** Algotester lab 7\_8 v3

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/algotester_lab_7_8_yana_lebedynska.cpp>

#include <iostream>

using namespace std;

template <class T>

class Node

{

public:

T data;

Node\* left;

Node\* right;

Node(T value) : data(value), left(nullptr), right(nullptr) {}

};

template <class T>

class BinaryTree

{

private:

Node<T>\* root;

bool contains(Node<T>\* node, T number)

{

if(node == nullptr)

{

return false;

}

if(number == node->data)

{

return true;

}

else if(number < node->data)

{

return contains(node->left, number);

}

else

{

return contains(node->right, number);

}

}

Node<T>\* insert(Node<T>\* node, T number)

{

if(node == nullptr)

{

return new Node<T>(number);

}

if(number < node->data)

{

node->left = insert(node->left, number);

}

else if(number > node->data)

{

node->right = insert(node->right, number);

}

return node;

}

void print(Node<T>\* node)

{

if(node != nullptr)

{

print(node->left);

cout << node->data << " ";

print(node->right);

}

}

int count(Node<T>\* node)

{

if(node == nullptr)

{

return 0;

}

else

{

return 1 + count(node->left) + count(node->right);

}

}

public:

BinaryTree() : root(nullptr) {} // за допомогою констурктора вказую, що при створенні нового дерева корінь буде вказувати спочатку на null pointer

void insert(T number)

{

root = insert(root, number);

}

bool contains(T number)

{

return contains(root, number);

}

void print()

{

print(root);

cout << endl;

}

int size()

{

return count(root);

}

};

int main()

{

int Q;

cin >> Q;

BinaryTree<int> tree;

while(Q--) // поки Q не стане нулем при кожній ітерації

{

string operation;

cin >> operation;

if(operation == "insert")

{

int number;

cin >> number;

tree.insert(number);

}

else if(operation == "contains")

{

int number;

cin >> number;

if(tree.contains(number))

cout << "Yes" << endl;

else

cout << "No" << endl;

}

else if(operation == "size")

{

cout << tree.size() << endl;

}

else if(operation == "print")

{

tree.print();

}

}

return 0;

}

## *код задачі algotester lab 7\_8 v3*

**Завдання №4** Class practice task 1

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/practice_class_task1_yana_lebedynska.cpp>

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {}

};

Node\* reverse(Node\* HEAD)

{

Node\* previous = nullptr;

Node\* next = nullptr;

Node\* current = HEAD;

while(current != nullptr)

{

next = current->next;

current->next = previous; //обертання (вказує не наступний вузол, а на попередній)

previous = current;

current = next;

}

return previous;

}

void printList(Node\* HEAD)

{

while(HEAD != nullptr)

{

cout << HEAD->data << " ";

HEAD = HEAD->next;

}

cout << endl;

}

int main()

{

Node\* HEAD = new Node(1);

HEAD->next = new Node(2);

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

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

cout << "The list BEFORE reversing: ";

printList(HEAD);

HEAD = reverse(HEAD);

cout << "The list AFTER reversing: ";

printList(HEAD);

return 0;

}

## *код задачі class practice task 1*

**Завдання №5** *class practice task 2*

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/practice_class_task2_yana_lebedynska.cpp>

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {};

};

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

{

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

{

if(h1->data != h2->data) //перевіряю на однаковість даних

{

return false;

}

h1 = h1->next;

h2 = h2->next;

}

return h1 == nullptr && h2 == nullptr; //перевіряю на довжину

}

int main()

{

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);

//HEAD2->next->next->next = new Node(4);

if(compare(HEAD1, HEAD2))

{

cout << "The lists are identical!";

}

else

{

cout << "The lists are different!";

}

return 0;

}

;

## *код задачі class practice task 2*

**Завдання №6** Class Practice Task 3

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/practice_class_task3_yana_lebedynska.cpp>

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {}

};

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

{

Node\* current = nullptr;

Node\* result = nullptr;

int extra = 0;

while (n1 != nullptr || n2 != nullptr || extra != 0)

{

int sum = extra;

if(n1 != nullptr)

{

sum += n1->data;

n1 = n1->next;

}

if(n2 != nullptr)

{

sum += n2->data;

n2 = n2->next;

}

extra = sum / 10;

sum %= 10;

Node\* newNode = new Node(sum);

if(result == nullptr)

{

result = newNode;

current = result;

}

else

{

current->next = newNode;

current = newNode;

}

}

return result;

}

void PrintReversedNum(Node\* num)

{

if(num == nullptr)

{

return;

}

PrintReversedNum(num->next);

cout << num->data;

}

int main()

{

Node\* num1 = new Node(9);

num1->next = new Node(7);

num1->next->next = new Node(3);

Node\* num2 = new Node(1);

num2->next = new Node(1);

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

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

cout << "Number 1: ";

PrintReversedNum(num1);

cout << endl;

cout << "Number 2: ";

PrintReversedNum(num2);

cout << endl;

cout << "The sum is: ";

PrintReversedNum(result);

cout << endl;

return 0;

}

## *код задачі class practice task 3*

**Завдання №7**  *class practice task 4*

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/practice_class_task4_yana_lebedynska.cpp>

#include <iostream>

using namespace std;

struct TreeNode

{

int data;

TreeNode\* left;

TreeNode\* right;

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

};

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

{

if(root == nullptr)

{

return nullptr;

}

TreeNode\* mirroredNode = new TreeNode(root->data);

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

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

return mirroredNode;

}

void CorrectOutput(TreeNode\* root)

{

if(root != nullptr)

{

CorrectOutput(root->left);

cout << root->data << " ";

CorrectOutput(root->right);

}

}

int main()

{

TreeNode\* root = new TreeNode(10);

root->left = new TreeNode(1);

root->left->right = new TreeNode(3);

root->left->right->left = new TreeNode(2);

root->right = new TreeNode(22);

root->right->left = new TreeNode(11);

cout << "Original binary tree: ";

CorrectOutput(root);

cout << endl;

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

cout << "Mirrored binary tree: ";

CorrectOutput(mirroredTree);

cout << endl;

return 0;

}

## *код задачі class practice task 4*

**Завдання №8**  *class practice task 5*

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/practice_class_task5_yana_lebedynska.cpp>

#include <iostream>

#include <stack>

using namespace std;

struct TreeNode

{

int data;

TreeNode\* left;

TreeNode\* right;

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

};

void tree\_sum(TreeNode\* root)

{

if(root == nullptr)

{

return;

}

stack<TreeNode\*> stackOfNode;

TreeNode\* current = root; //вказівник на поточний вузол

TreeNode\* previous = nullptr; //для відстеження попереднього вузла

while(current != nullptr || !stackOfNode.empty())

{

while(current != nullptr)

{

stackOfNode.push(current);

current = current->left;

}

current = stackOfNode.top();

if(current->left == nullptr && current->right == nullptr)

{

stackOfNode.pop();

previous = current;

current = nullptr;

}

else if(current->right != nullptr && current->right != previous) //якщо є правий вузол, який ще не відвіданий

{

current = current->right;

}

else

{

TreeNode\* temp = current;

current = nullptr;

stackOfNode.pop(); //видалити поточний вузол зі стеку, так як він вже оброблений

int sum = temp->data;

if(temp->left != nullptr)

{

sum += temp->left->data;

}

if(temp->right != nullptr)

{

sum += temp->right->data;

}

temp->data = sum;

previous = temp; //щоб на наступній ітерації циклу правильно визначити, чи було відвідане праве піддерево

}

}

}

void printBinaryTree(TreeNode\* root)

{

if(root != nullptr)

{

printBinaryTree(root->left);

cout << root->data << " ";

printBinaryTree(root->right);

}

}

int main()

{

TreeNode\* root = new TreeNode(5);

root->left = new TreeNode(3);

root->left->left = new TreeNode(1);

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

root->right = new TreeNode(8);

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

root->right->right = new TreeNode(10);

cout << "Original binary tree: ";

printBinaryTree(root);

cout << endl;

tree\_sum(root);

cout << "Tree after sums: ";

printBinaryTree(root);

cout << endl;

return 0;

}

## *код задачі class practice task 5*

**Завдання №8**  *self practice (algotester)*

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/blob/19ea47eb24c5cba962322736e5741b0a3b302605/ai_12/yana_lebedynska/Epic6/practice_self_task_yana_lebedynska.cpp>

#include <iostream>

using namespace std;

int main() {

long n;

cin >> n;

string results;

cin >> results;

long kScores = 0, vScores = 0;

long kGames = 0, vGames = 0;

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

if (results[i] == 'K') {

kScores++;

} else {

vScores++;

}

if ((kScores >= 11 || vScores >= 11) && abs(kScores - vScores) >= 2) {

if (kScores > vScores) {

kGames++;

} else {

vGames++;

}

kScores = 0;

vScores = 0;

}

}

cout << kGames << ":" << vGames << endl;

if (kScores > 0 || vScores > 0)

{

cout << kScores << ":" << vScores << endl;

}

return 0;

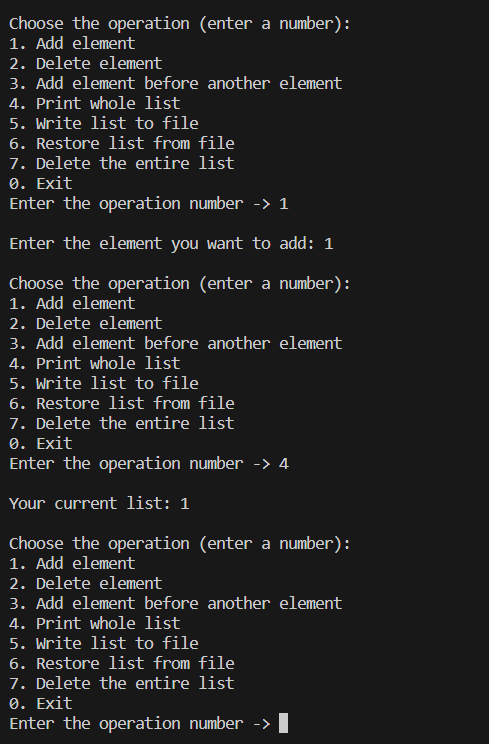
}

## *код задачі self practice (algotester)*

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

***Завдання №1*** VNS Lab 10

**Результат програми:**

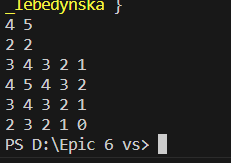
****

*робота програми*

Час затрачений на виконання: 4 год

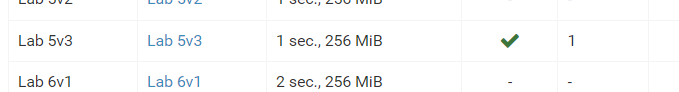
***Завдання №2*** Algotester lab 5

**Робота програми:**

****

*результат програми*

**Evidences:**

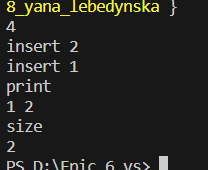
****

****

Час затрачений на виконання завдання: 1 год 30 хв

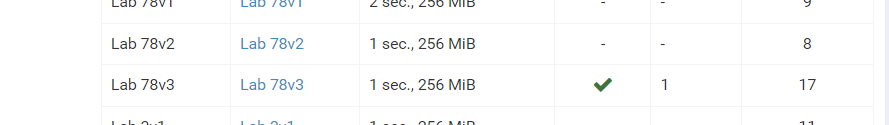
***Завдання №3*** Algotester lab 7\_8

**Результат програма:**

****

**Evidences:**

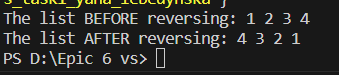
****

****

Час затрачений на виконання завдання: 2 год

***Завдання №4*** Class practice task 1

**Робота програми:**

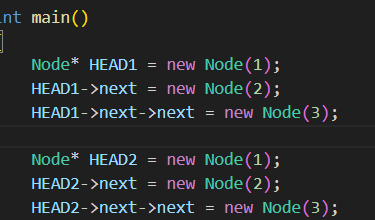


Час затрачений на виконання завдання: 1 год

***Завдання №5*** Class practice task 2

**Робота програми:**

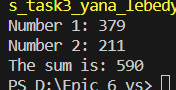
****

****

Час затрачений на виконання: 1 год

***Завдання №6*** Class Practice Task 3

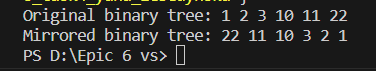
**Результат програми:**

****

Час затрачений на виконання: 1 год

***Завдання №7*** Class Practice Task 4

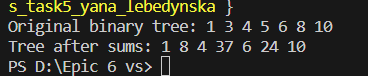
**Результат програми:**

****

Час затрачений на виконання: 40 хв

***Завдання №8*** Class Practice Task 5

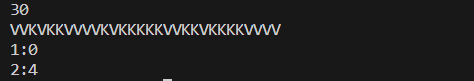
**Результат програми:**

****

Час затрачений на виконання: 40 хв

***Завдання №9*** Self practice (Algotester)

**Результат програми:**



# **Evidences:**

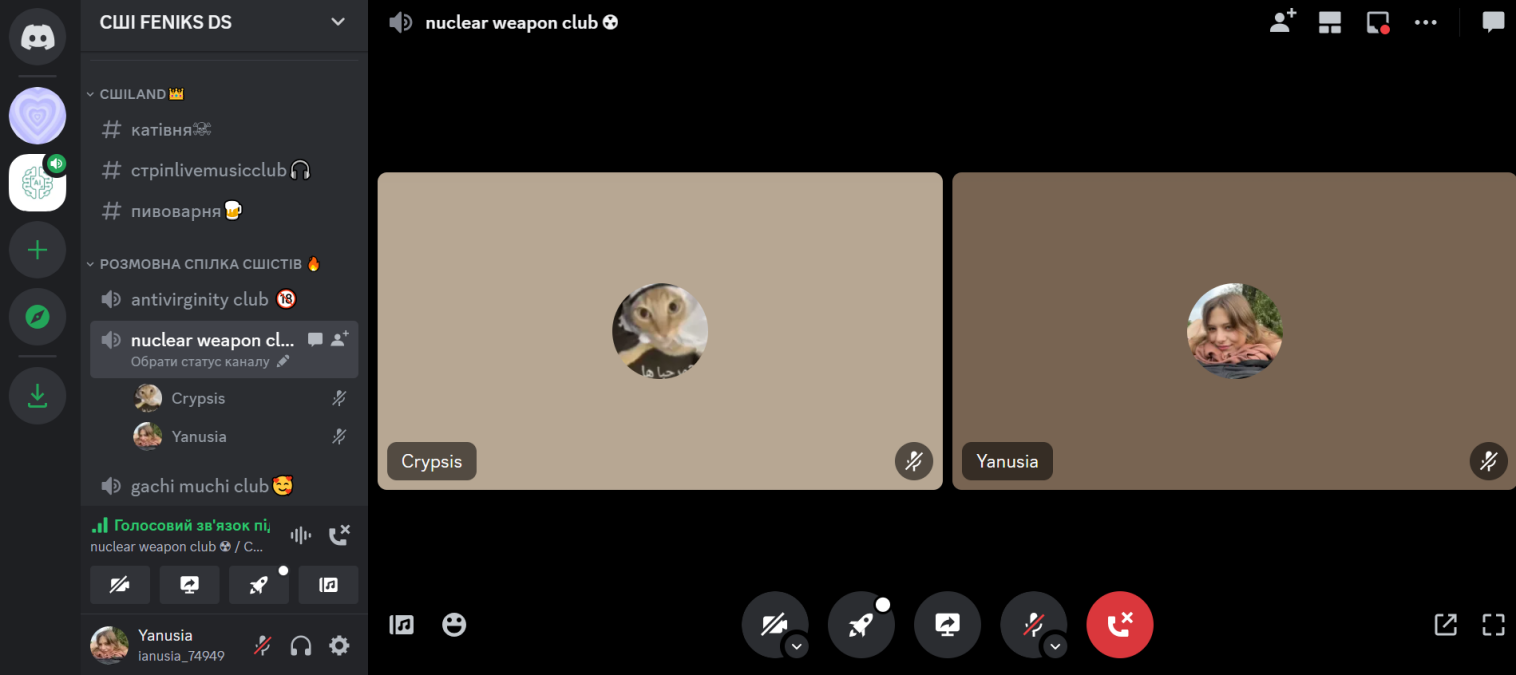


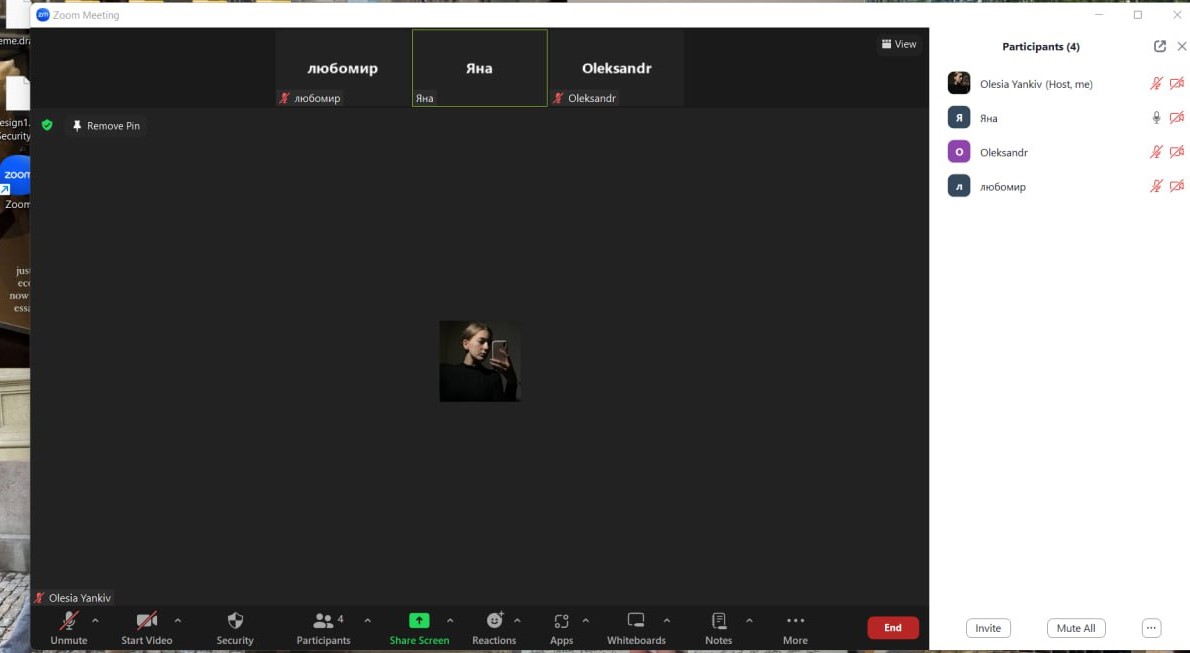
# 

# 

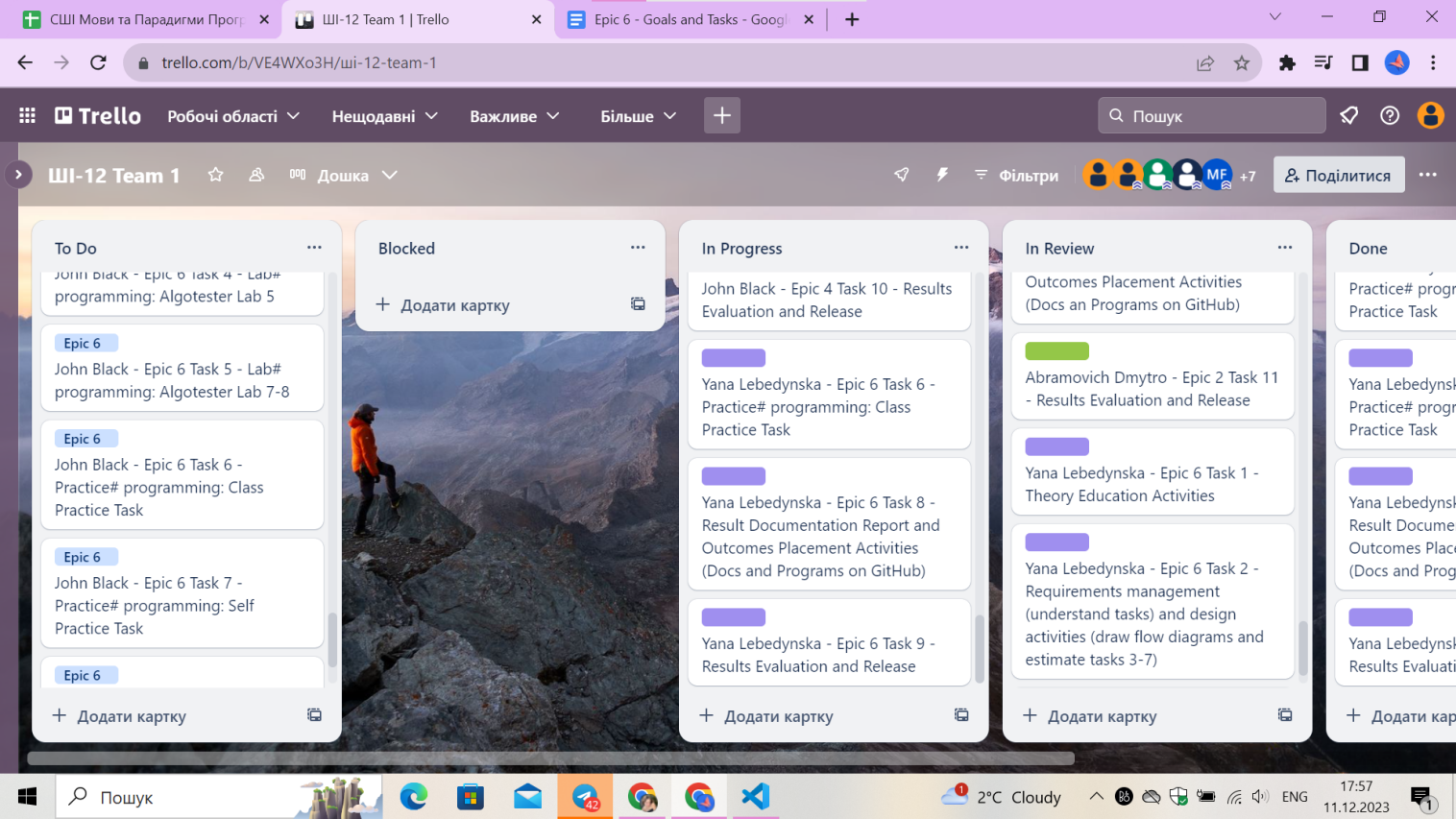
# **Робота команди:**

**Міти**

****

****

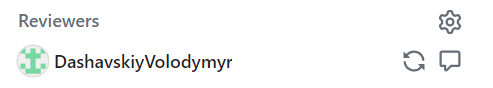
**Трелло**

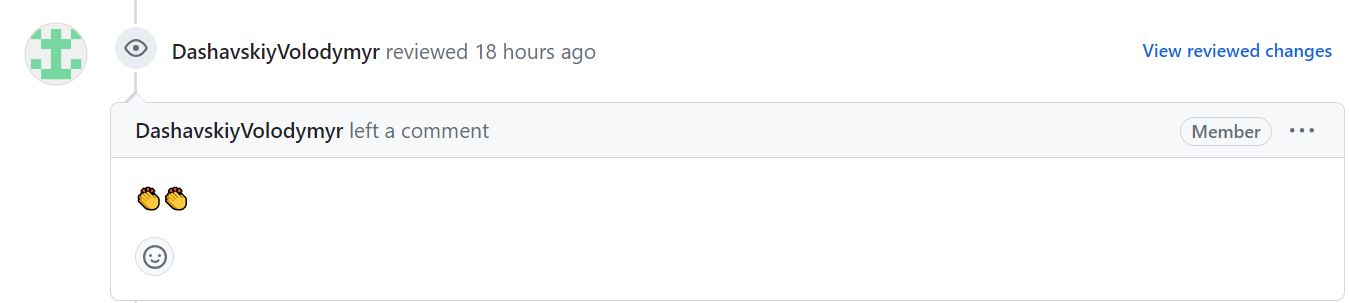
****

*скрін робочої області в трелло*

**Пул-реквест:**

**<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/790>**

****

****

*коментар до пул-реквесту в гітХабі*

**Висновки:**

Протягом епіка 6 я змогла набути навиків у роботі з однозв’язними і двозв’язними списками, ознайомилась і навчилась працювати здеревами, попрактикувалась у роботі зі стеком. Практично закріпила всі новонабуті навики.