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, Дерево). Алгоритми обробки дерев.»

***Виконала:***

студент групи ШІ-12

Пасічко Софія Андріївна

# **Тема роботи:**

Ознайомлення з динамічними структурами – чергою, стеком, деревом. Опрацювання однозв’язних та двозв’язних списків. Вивчення алгоритмів обробки дерев.

# **Мета роботи:**

Теоретично та практично ознайомитися з динамічними структурами та алгоритмами обробки дерев.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури. Черга. Стек.
* Тема №2: Динамічні структури. Зв’язні списки.
* Тема №3: Динамічні структури. Дерево.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури. Черга. Стек.
  + Джерела Інформації
  + <https://www.youtube.com/watch?v=Yhw8NbjrSFA>
  + <https://www.youtube.com/watch?v=NyOjKd5Qruk>
  + Що опрацьовано:
    - Ознайомлена із чергою. Різниця між чергою та стеком.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12
  + Звершення опрацювання теми: 07.12
* Тема №2: Динамічні структури. Зв’язні списки.
  + Джерела Інформації:
  + <https://www.youtube.com/watch?v=-25REjF_atI>
  + <https://www.youtube.com/watch?v=QLzu2-_QFoE&t=648s>
  + Що опрацьовано:
    - Однозв’язний список. Двозв’язний список.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12
  + Звершення опрацювання теми: 08.12
* Тема №3 Динамічні структури. Дерево. Алгоритми обробки дерев.
  + Джерела Інформації:
  + <https://www.youtube.com/watch?v=qBFzNW0ALxQ>
  + <http://elcat.pnpu.edu.ua/docs/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8%20%D1%96%20%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85/lab8-9_tree.html>
  + Що опрацьовано:
    - Поняття бінарного дерева. Методи їх обробки.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12
  + Звершення опрацювання теми: 10.12

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10.

Варіант 13.

1. Написати функцію для створення списку. Функція може створювати

порожній список, а потім додавати в нього елементи.

2. Написати функцію для друку списку. Функція повинна передбачати вивід

повідомлення, якщо список порожній.

3. Написати функції для знищення й додавання елементів списку у

відповідності зі своїм варіантом.

4. Виконати зміни в списку й друк списку після кожної зміни.

5. Написати функцію для запису списку у файл.

6. Написати функцію для знищення списку.

7. Записати список у файл, знищити його й виконати друк (при друці повинне

бути видане повідомлення "Список порожній").

8. Написати функцію для відновлення списку з файлу.

9. Відновити список і роздрукувати його.

10.Знищити список.

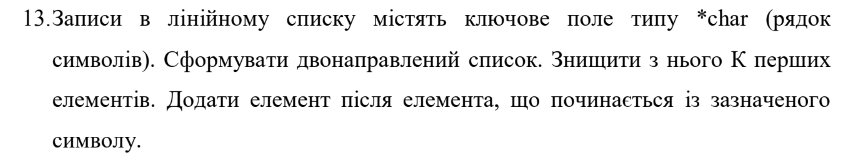


Рисунок . Умова завдання 1.

Завдання №2 Аlgotester Lab 5.

Варіант 3.

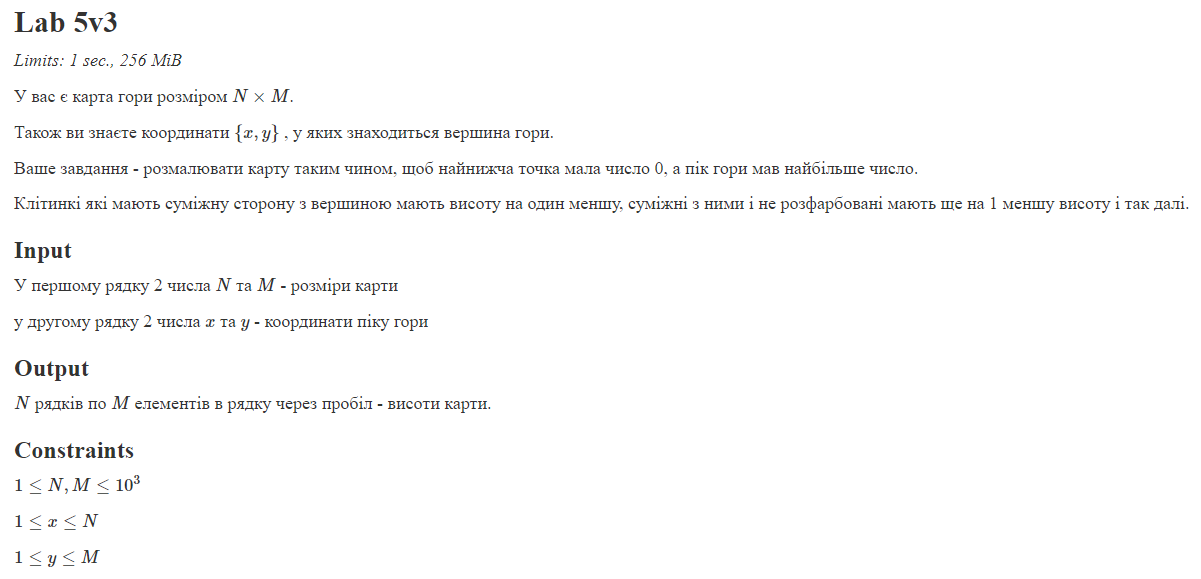


Рисунок . Умова завдання 2.

Важливо врахувати обмеження під час імплементації програми.

Завдання №3 Algotester Lab 7-8.

Варіант 3.



Рисунок . Умова завдання 3

Важливо врахувати обмеження під час імплементації програми.

Завдання №4 Class Practice Task 1.

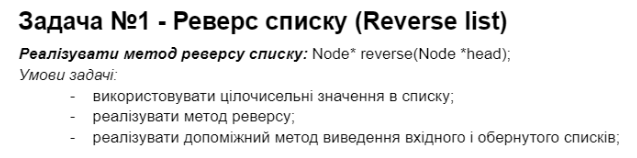


Рисунок . Умова завдання 4.

Завдання №5 Class Practice Task 2.





Рисунок . Умова завдання 5.

Завдання №6 Class Practice Task 3.



Рисунок . Умова завдання 6.

Завдання №7 Class Practice Task 4.

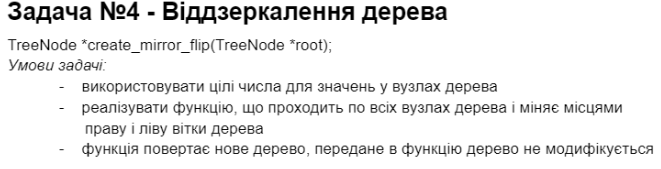


Рисунок . Умова завдання 7.

Завдання №8 Class Practice Task 5.

## **Задача №5 - Записати кожному батьківському вузлу суму підвузлів**

void tree\_sum(TreeNode \*root);

*Умови задачі:*

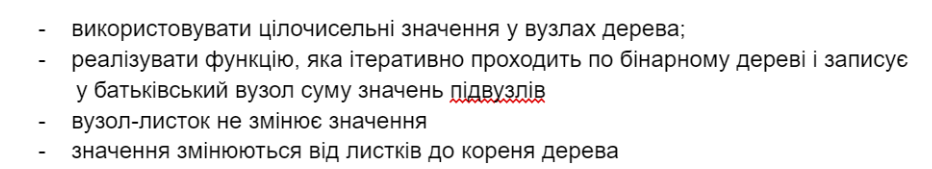


Рисунок . Умова завдання 8.

Завдання №9 Self-Practice Task.

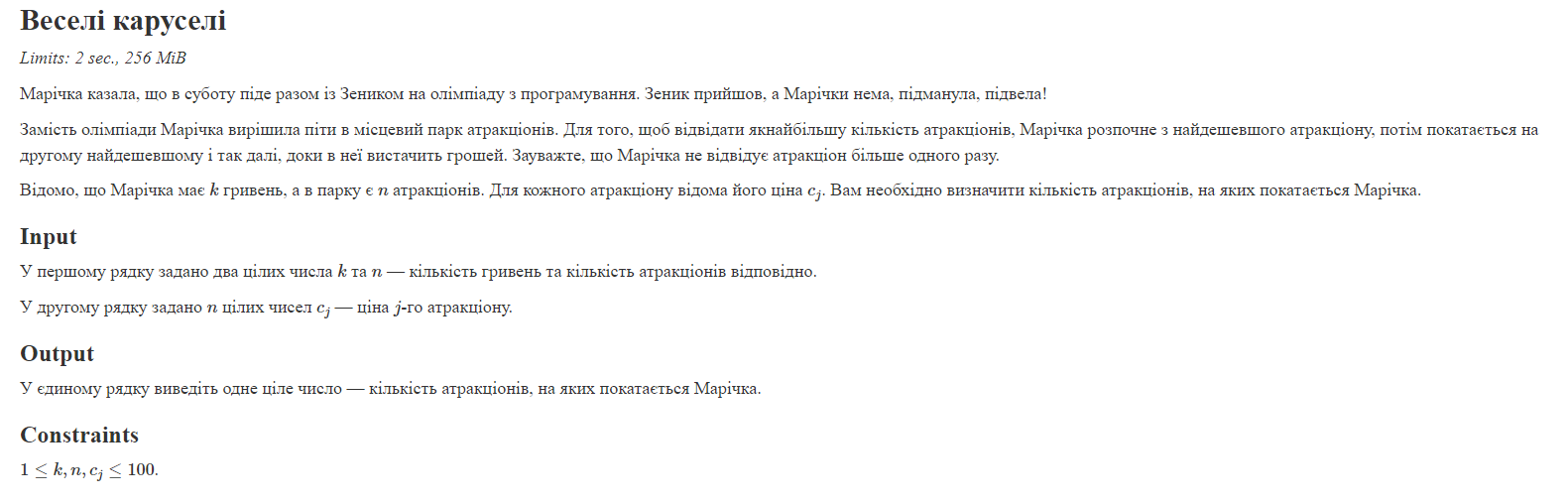


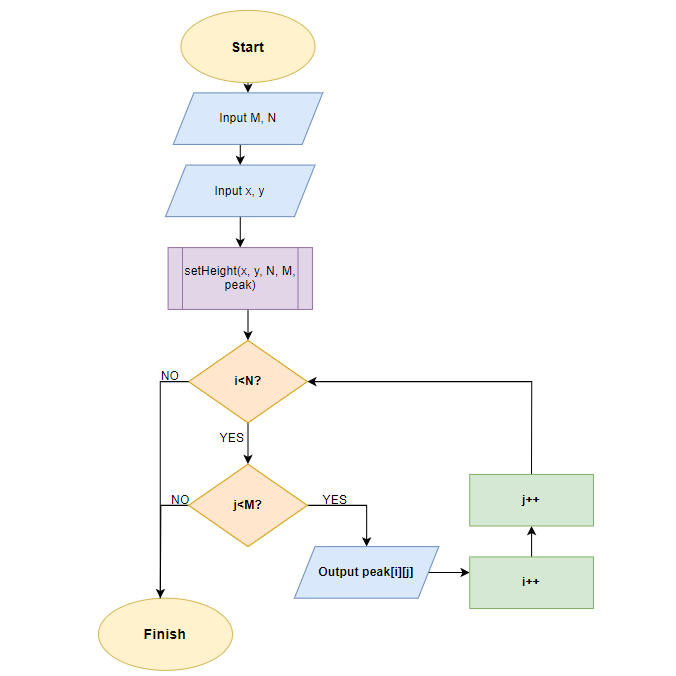
Рисунок 9. Умова завдання 9.

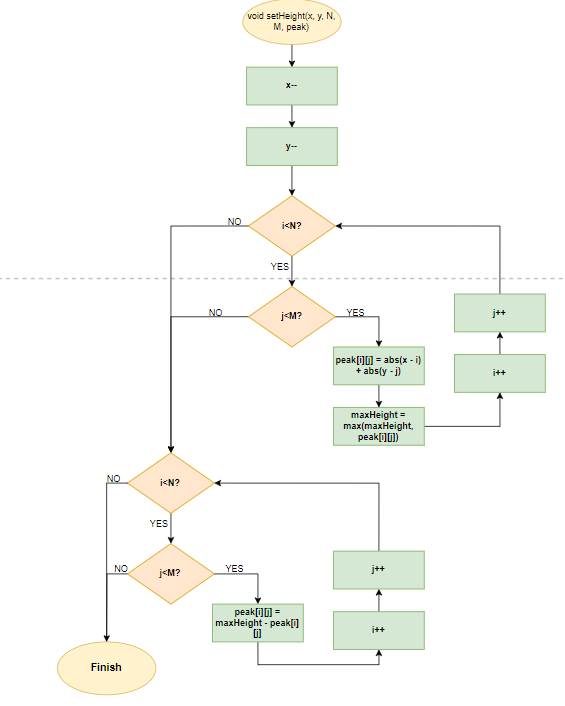
## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 VNS Lab 10.

Планований час на виконання – 2 год

Програма №2 Аlgotester Lab 5.





Планований час на виконання – 2 год

Програма №3 Algotester Lab 7-8.

Планований час на виконання – 2 год

Програма №4 Class Practice Task 1.

Планований час на виконання – 1 год

Програма №5 Class Practice Task 2.

Планований час на виконання – 1 год

Програма №6 Class Practice Task 3.

Планований час на виконання – 1 год

Програма №7 Class Practice Task 4.

Планований час на виконання – 1 год

Програма №8 Class Practice Task 5.

Планований час на виконання – 1,5 год

Програма №9 Self-Practice Task.

Планований час на виконання – 20 хв

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

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

Завдання №1

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

struct Node {

    char\* data;

    Node\* next;

    Node\* prev;

    Node(const char\* str) {

        data = new char[strlen(str) + 1];

        strcpy(data, str);

        next = nullptr;

        prev = nullptr;

    }

};

Node\* createList() {

    return nullptr;

}

void printList(Node\* head) {

    if (head == nullptr) {

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

        return;

    } else {

        Node\* current = head;

        while (current != nullptr) {

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

            current = current->next;

        }

        cout << endl;

    }

}

Node\* addElement(Node\* head, const char\* str, char symbol) { //додавання елементів після заданого символу

    Node\* newNode = new Node(str);

    if (head == nullptr) {

        return newNode;

    }

    Node\* current = head;

    while (current != nullptr && current->data[0] != symbol) {

        current = current->next;

    }

    if (current != nullptr) {

        newNode->next = current->next;

        newNode->prev = current;

        if (current->next != nullptr) {

            current->next->prev = newNode;

        }

        current->next = newNode;

    }

    return head;

}

Node\* addNode(Node\* head, const char\* str) { //додавання елементів у кінець списку

    Node\* newNode = new Node(str);

    if (head == nullptr) {

        return newNode;

    }

    Node\* current = head;

    while (current->next != nullptr) {

        current = current->next;

    }

    current->next = newNode;

    newNode->prev = current;

    return head;

}

Node\* removeKElements(Node\* head, int K) {

    for (int i = 0; i < K && head != nullptr; ++i) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

    if (head != nullptr) {

        head->prev = nullptr;

    }

    return head;

}

void destroyList(Node\*& head) {

    while (head != nullptr) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

}

void writeToFile(Node\* head, const string& filename) {

    ofstream file(filename);

    if (!file) {

        cout << "Failure" << endl;

        return;

    }

    while (head != nullptr) {

        file << head->data << " ";

        head = head->next;

    }

    file.close();

}

Node\* removeKFirstElements(Node\* head, int k) {

    for (int i = 0; i < k && head != nullptr; ++i) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

    if (head != nullptr) {

        head->prev = nullptr;

    }

    return head;

}

Node\* restoreFromFile(const string& filename) {

    ifstream file(filename);

    if (!file) {

        cout << "Failure" << endl;

        return nullptr;

    }

    Node\* head = nullptr;

    char str[256];

    while (file >> str) {

        head = addNode(head, str);

    }

    file.close();

    return head;

}

int main() {

    Node\* newList = createList();

    // Додавання елементів та друк списку

    newList = addNode(newList, "Monday");

    newList = addNode(newList, "Tuesday");

    newList = addNode(newList, "Friday");

    printList(newList);

    cout << endl;

    newList = removeKFirstElements(newList, 1);

    printList(newList);

    cout << endl;

    writeToFile(newList, "file.txt");

    newList = addElement(newList, "Wednesday", 'T');

    printList(newList);

    cout << endl;

    writeToFile(newList, "file.txt");

    destroyList(newList);

    printList(newList);

    cout << endl;

    newList = restoreFromFile("file.txt");

    printList(newList);

    cout << endl;

    destroyList(newList);

    printList(newList);

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-39c1fec77c6a7cd9cd9a1a8e070e36af265c3d937e6be4aa03cce75b2a09a93d>

Завдання №2

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

void setHeight(int x, int y, int N, int M, vector<vector<int>>& peak) {

    int maxHeight = 0;

    x--;

    y--;

    for (int i = 0; i < N; ++i) {

        for (int j = 0; j < M; ++j) {

            peak[i][j] = abs(x - i) + abs(y - j);

            maxHeight = max(maxHeight, peak[i][j]);

        }

    }

    for (int i = 0; i < N; ++i) {

        for (int j = 0; j < M; ++j) {

            peak[i][j] = maxHeight - peak[i][j];

        }

    }

}

int main() {

    int N, M, x, y;

    cin >> N >> M;

    cin >> x >> y;

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

    setHeight(x, y, N, M, peak);

    for (int i = 0; i < N; ++i) {

        for (int j = 0; j < M; ++j) {

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

        }

        cout << endl;

    }

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-aee1dcf6927262b8fb4a436dea9b3f80c866341803a815f4deea36811db48979>

Завдання №3

#include <iostream>

using namespace std;

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

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

};

TreeNode\* createNewTree() {

    return nullptr;

}

TreeNode\* insert(TreeNode\* root, int number) {

    if (root == nullptr) {

        return new TreeNode(number);

    }

    if (number < root->data) {

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

    } else if (number > root->data) {

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

    }

    return root;

}

bool contains(TreeNode\* node, int number) {

    while (node != nullptr) {

        if (number == node->data) {

            return true;

        } else if (number < node->data) {

            node = node->left;

        } else {

            node = node->right;

        }

    }

    return false;

}

int size(TreeNode\* node) {

    if (node == nullptr) {

        return 0;

    }

    int leftSize = size(node->left);

    int rightSize = size(node->right);

    int size = 1 + leftSize + rightSize;

    return size;

}

void print(TreeNode\* node) {

    if (node != nullptr) {

        print(node->left);

        cout << node->data << " ";

        print(node->right);

    }

}

int main() {

    int Q;

    cin >> Q;

    TreeNode\* root = createNewTree();

    for (int i = 0; i < Q; i++){

        string option;

        cin >> option;

        if (option == "insert") {

            int number;

            cin >> number;

            root = insert(root, number);

        } else if (option == "contains") {

            int number;

            cin >> number;

            if (contains(root, number)==true) {

                cout << "Yes"<< endl;

            } else {

                cout << "No" << endl;

            }

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

            cout << size(root) << endl;

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

            print(root);

            cout << endl;

        }

    }

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-dbaee12802ea05d3d2265302c2c98e4632c81b8a57b65c89cc7ba56dc6166e66>

Завдання №4

#include <iostream>

#include <cstdlib>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

void printList(Node\* head) {

    while (head != nullptr) {

        cout << head->data << " ";

        head = head->next;

    }

    cout << endl;

}

void deleteList(Node\* head) {

    Node\* current = head;

    Node\* nextNode = nullptr;

    while (current != nullptr) {

        nextNode = current->next;

        delete current;

        current = nextNode;

    }

}

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;

    }

    head = prev;

    return head;

}

int main() {

    Node\* head = new Node{7, nullptr};

    head->next = new Node{1, nullptr};

    head->next->next = new Node{2, nullptr};

    head->next->next->next = new Node{0, nullptr};

    head->next->next->next->next = new Node{3, nullptr};

    head = reverse (head);

    printList(head);

    deleteList(head);

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-550865d323435006d3b9495fc4ae6ef6caf8b464cd58694e13ca191512681bad>

Завдання №5

#include <iostream>

#include <cstdlib>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

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

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

        if (h1->data != h2->data) {

            return false;

        }

        h1 = h1->next;

        h2 = h2->next;

    }

    if (h1 == nullptr && h2 == nullptr) {

        return true;

    }

    return false;

}

int main() {

    Node\* h1 = new Node{7, nullptr};

    h1->next = new Node{1, nullptr};

    h1->next->next = new Node{2, nullptr};

    h1->next->next->next = new Node{0, nullptr};

    h1->next->next->next->next = new Node{3, nullptr};

    Node\* h2 = new Node{7, nullptr};

    h2->next = new Node{1, nullptr};

    h2->next->next = new Node{2, nullptr};

    h2->next->next->next = new Node{0, nullptr};

    h2->next->next->next->next = new Node{3, nullptr};

    bool comparison = compare(h1, h2);

    if (comparison=true) {

        cout << "Lists are similar" << endl;

    } else {

        cout << "Lists are different" << endl;

    }

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-81dac01576158f0177d5e782ad8d6d31cd67d148e43c88ac5d2fd8862819af76>

Завдання №6

#include <iostream>

#include <vector>

#include <cstdlib>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

vector<int> listIntoVector(Node\* head) {

    vector<int> result;

    while (head != nullptr) {

        result.push\_back(head->data);

        head = head->next;

    }

    return result;

}

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

    vector<int> firstNumber = listIntoVector(n1);

    vector<int> secondNumber = listIntoVector(n2);

    int maxLength = max(firstNumber.size(), secondNumber.size());

    firstNumber.resize(maxLength, 0);

    secondNumber.resize(maxLength, 0);

    int a=0;

    vector<int> result;

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

        int sum = firstNumber[i] + secondNumber[i] + a;

        result.push\_back(sum % 10);

        a = sum / 10;

    }

    if (a > 0) {

        result.push\_back(a);

    }

    Node\* head = nullptr;

    Node\* tail = nullptr;

    for (int number : result) {

        Node\* newNode = new Node{number, nullptr};

        if (head == nullptr) {

            head = newNode;

            tail = newNode;

        } else {

            tail->next = newNode;

            tail = newNode;

        }

    }

    return head;

}

void printList(Node\* head) {

    while (head != nullptr) {

        cout << head->data;

        if (head->next != nullptr) {

            cout << " ";

        }

        head = head->next;

    }

    cout << endl;

}

int main() {

    Node\* firstNumber = new Node{1, nullptr};

    firstNumber->next = new Node{4, nullptr};

    firstNumber->next->next = new Node{5, nullptr};

    Node\* secondNumber = new Node{3, nullptr};

    secondNumber->next = new Node{0, nullptr};

    secondNumber->next->next = new Node{2, nullptr};

    secondNumber->next->next->next = new Node{7, nullptr};

    Node\* sum = add(firstNumber, secondNumber);

    printList(sum);

    while (firstNumber != nullptr) {

        Node\* temp = firstNumber;

        firstNumber = firstNumber->next;

        delete temp;

    }

    while (secondNumber != nullptr) {

        Node\* temp = secondNumber;

        secondNumber = secondNumber->next;

        delete temp;

    }

    while (sum != nullptr) {

        Node\* temp = sum;

        sum = sum->next;

        delete temp;

    }

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-9eb21f19f9739f306f30a19511d96c48173c93789383ff60eb6d0d4d7abc4fab>

Завдання №7

#include <iostream>

#include <queue>

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) {

    if (root == nullptr) {

        return nullptr;

    }

    TreeNode\* mirror\_flip\_node = new TreeNode(root->data);

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

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

    return mirror\_flip\_node;

}

void print\_tree(TreeNode \*root)

{

    if (root != NULL) {

        print\_tree(root->left);

        cout << root->data <<" ";

        print\_tree(root->right);

    }

    return;

}

int main() {

    TreeNode\* root = new TreeNode(10);

    root->left = new TreeNode(5);

    root->right = new TreeNode(11);

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

    root->right->right = new TreeNode(17);

    TreeNode\* mirrored\_tree = create\_mirror\_flip(root);

    print\_tree(root);

    cout << endl;

    print\_tree(mirrored\_tree);

    delete root;

    delete mirrored\_tree;

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-2ff80217d534c0d0539ff521fd54715cd4062326e851c7451e8eb50adfbcc7ef>

Завдання №8

#include <iostream>

#include <stack>

using namespace std;

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

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

};

void tree\_sum(TreeNode\* root) {

    if (root == nullptr) {

        return;

    }

    stack<TreeNode\*> nodes;

    TreeNode\* current = root;

    int sum = 0;

    while (current != nullptr || !nodes.empty()) {

        while (current != nullptr) {

            nodes.push(current);

            current = current->left;

        }

        current = nodes.top();

        nodes.pop();

        sum += current->data;

        current->data = sum;

        current = current->right;

    }

}

void print\_tree(TreeNode \*root)

{

    if (root != NULL) {

        print\_tree(root->left);

        cout << root->data <<" ";

        print\_tree(root->right);

    }

    return;

}

void delete\_tree(TreeNode\* root) {

    if (root != nullptr) {

        delete\_tree(root->left);

        delete\_tree(root->right);

        delete root;

    }

}

int main() {

    TreeNode\* root = new TreeNode(10);

    root->left = new TreeNode(8);

    root->right = new TreeNode(15);

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

    root->left->right = new TreeNode(9);

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

    root->right->right = new TreeNode(17);

    print\_tree(root);

    cout << endl;

    tree\_sum(root);

    print\_tree(root);

    cout << endl;

    delete\_tree(root);

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-cbc7f7f6929cf2bb00d4213d696245a8121e207d0f0451ae23b9f0681c7ce3a7>

Завдання №9

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

    int k, n;

    cin >> k >> n;

    int \*prices = new int[n];

    for (int i = 0; i < n; ++i) {

        cin >> prices[i];

    }

    sort(prices, prices+n);

    int numberOfAttractions=0;

    for (int i = 0; i < n; ++i) {

        k=k-prices[i];

        if (k>=0) {

            numberOfAttractions++;

        }

        else {

            k=k+prices[i];

        }

    }

    cout << numberOfAttractions;

    delete[] prices;

    return 0;

}



<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973/files#diff-7364243ce0d5197edf2d69db943b7601871544e5ceb570045c890b41f1a964e6>

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

Завдання №1

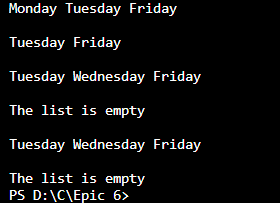


Рисунок 9. Результат виконання програми 1.

Час затрачений на виконання завдання – 2 год

Завдання №2

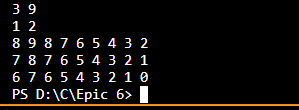
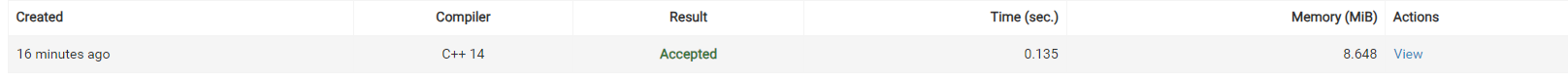


Рисунок 10. Результат виконання програми 2.



Час затрачений на виконання завдання – 2 год

Завдання №3

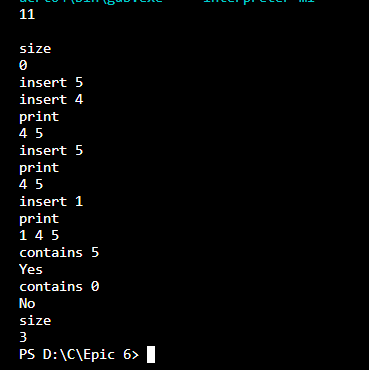
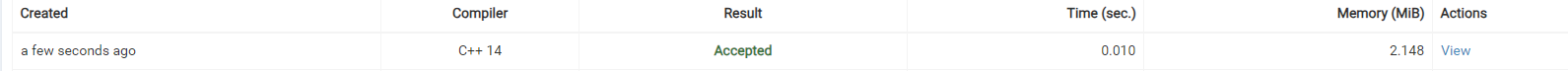


Рисунок 11. Результат виконання програми 3.



Час затрачений на виконання завдання – 3 год

Завдання №4



Рисунок 12. Результат виконання програми 4.

Час затрачений на виконання завдання – 1 год

Завдання №5



Рисунок 13. Результат виконання програми 5.

Час затрачений на виконання завдання – 1 год

Завдання №6



Рисунок 14. Результат виконання програми 6.

Час затрачений на виконання завдання – 1 год

Завдання №7



Рисунок 15. Результат виконання програми 7.

Час затрачений на виконання завдання – 1,5 год

Завдання №8

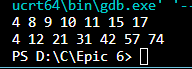


Рисунок 16. Результат виконання програми 8.

Час затрачений на виконання завдання – 1,5 год

Завдання №9

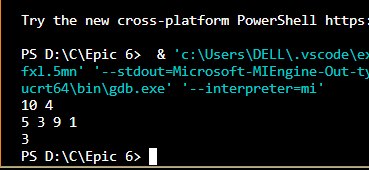
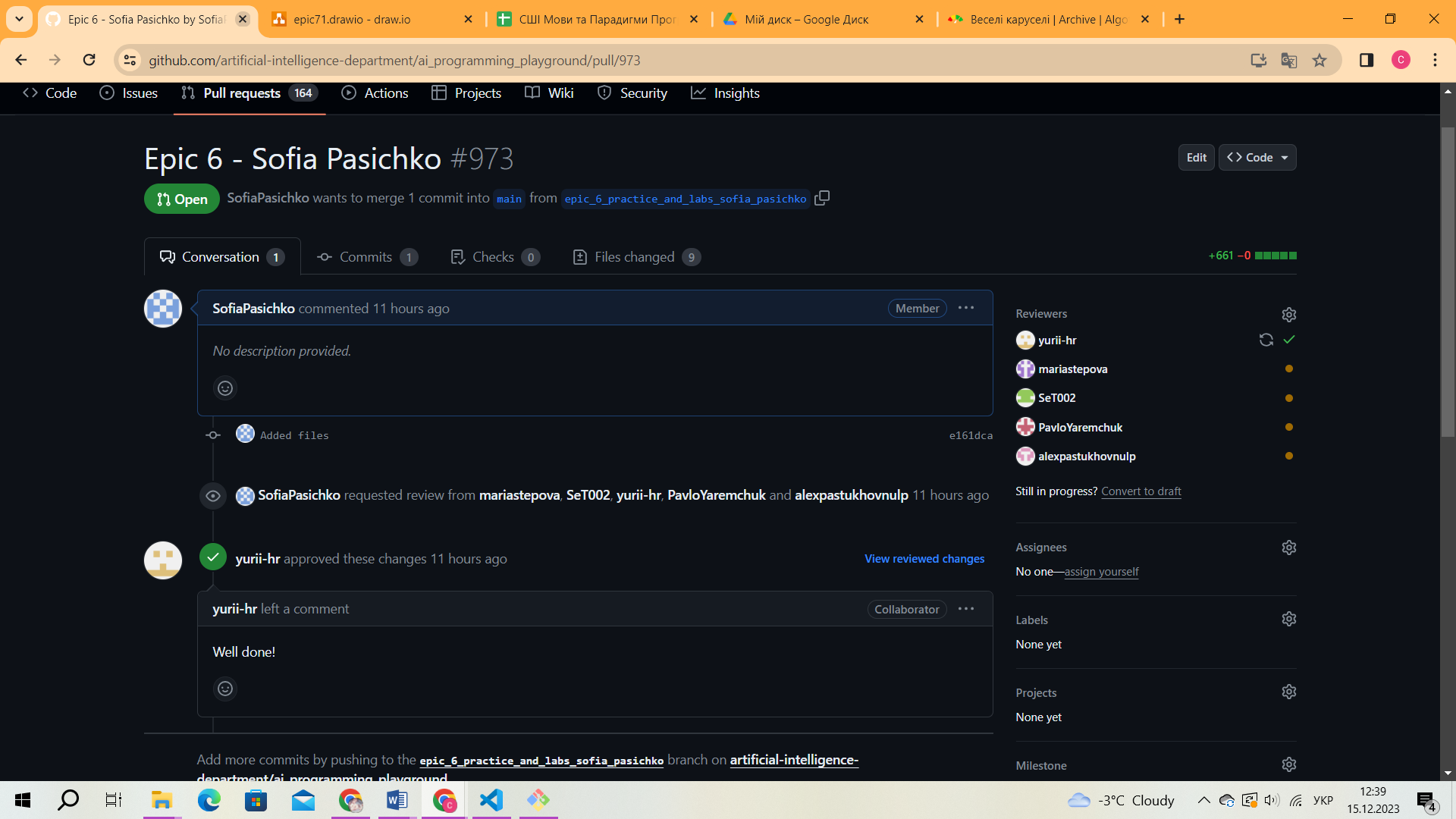


Рисунок . Результат виконання програми 9.

Час затрачений на виконання завдання – 20 хв

## **6. Кооперація з командою:**





# **Висновки:**

Протягом епіку 6 ознайомилася з динамічними структурами. Опрацювала зв’язні списки – однозв’язні та двозв’язні. Практично та теоретично опрацювала використання бінарних дерев.

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/973>