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-12

Стецюк Любомир Васильович

**Тема роботи:**

Ознайомлення з динамічними структурами даних, такими як черга, стек, списки та дерева, передбачає вивчення їхніх унікальних властивостей, функцій і особливостей використання.

**Мета роботи:**

Ознайомлення з основними динамічними структурами даних, такими як стеки, черги, списки і дерева, включає в себе зрозуміння їхніх особливостей, роботи та функцій. Вивчення операцій над цими структурами включає в себе розуміння способів додавання, вилучення та обробки даних, що зберігаються у цих структурах. Практичні навички реалізації алгоритмів з використанням динамічних структур даних в мові програмування C++ допомагають закріпити ці знання та вміння у практичному застосуванні.

**Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури (Черга, Стек, Списки, Дерево).
* Тема №2: Алгоритми обробки динамічних структур.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури (Черга, Стек, Списки, Дерево).
  + Джерела Інформації
    - Стаття.

[https://acode.com.ua/urok-111-stek-i-kupa/#toc-1](https://acode.com.ua/urok-111-stek-i-kupa/%23toc-1)

<https://www.bitdegree.org/learn/linked-list-c-plus-plus>

* + - Відео.

<https://www.youtube.com/watch?v=UHxtjVsOTHc>

* + Що опрацьовано:
    - Опрацьовано різні типи динамічних структур таких як черга, стек, списки(однозв’язний та двозв’язний), та дерево. Опрацьовано як вони виглядають і як використовуються.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 06.12.2023
  + Звершення опрацювання теми: 15.12.2023
* Тема №2: Алгоритми обробки динамічних структур.
  + Джерела Інформації:
    - Стаття.

<https://prometheus.org.ua/cs50/sections/section6.html>

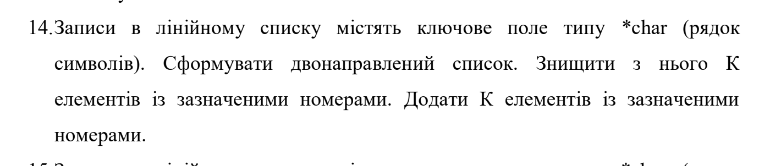
* + Що опрацьовано:
    - Розглянуто базові алгоритми обробки динамічних структур.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 06.12.2023
  + Звершення опрацювання теми: 15.12.2023

**Виконання роботи:**

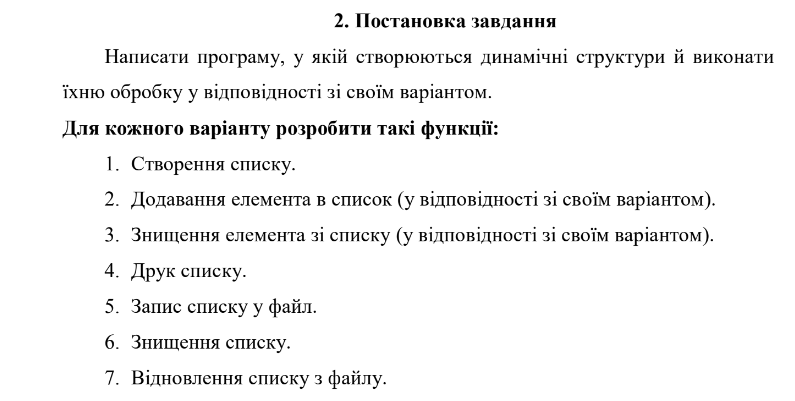
**1. Опрацювання завдання та вимог до програм та середовища:**

Завдання № 1. VNS Lab 10 – Task 1

* Варіант завдання:

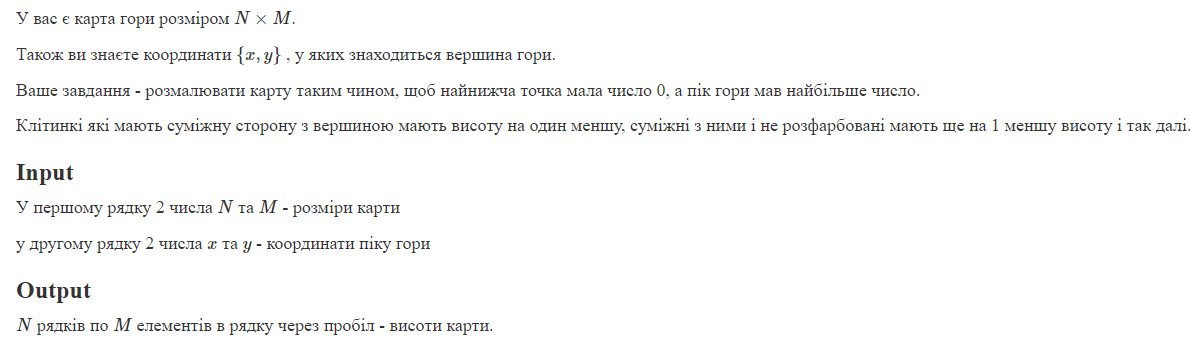


* Деталі завдання:



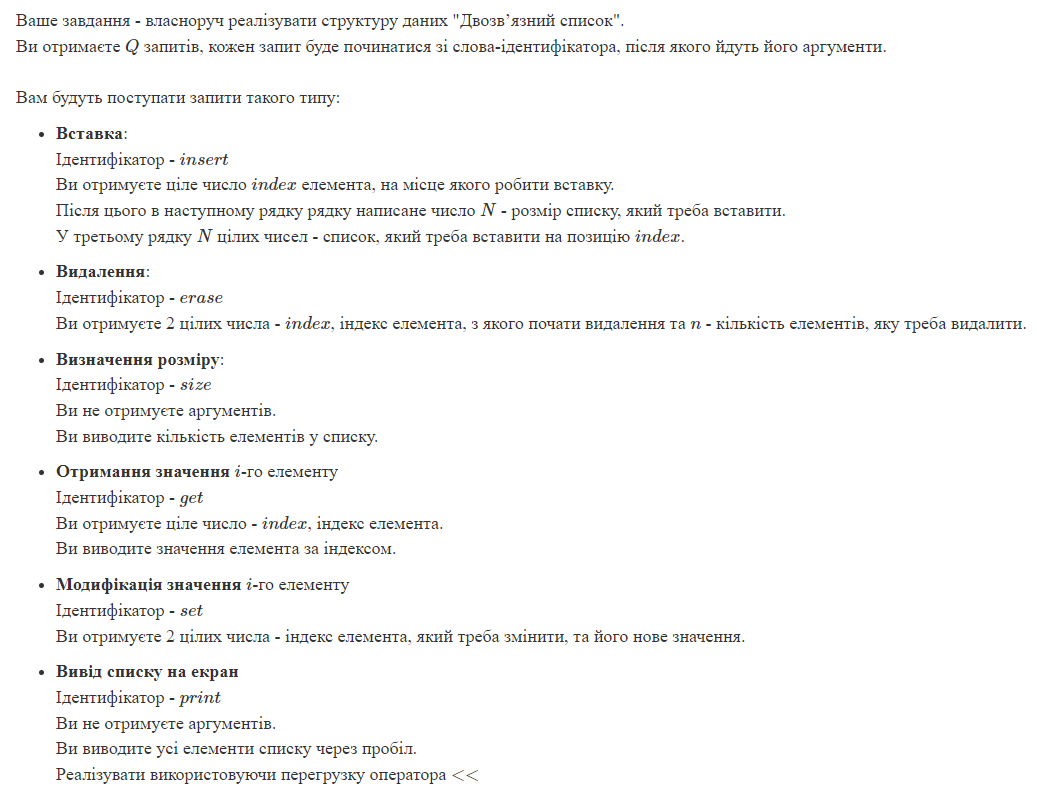
Завдання № 2. Algotester Lab 5 – Task 1

* Варіант завдання: 3
* Деталі завдання:



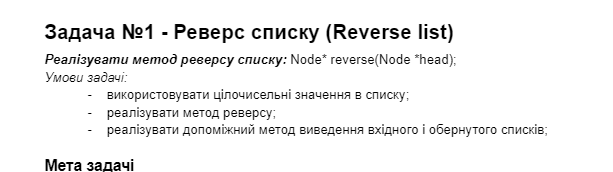
Завдання № 3. Algotester Lab 78

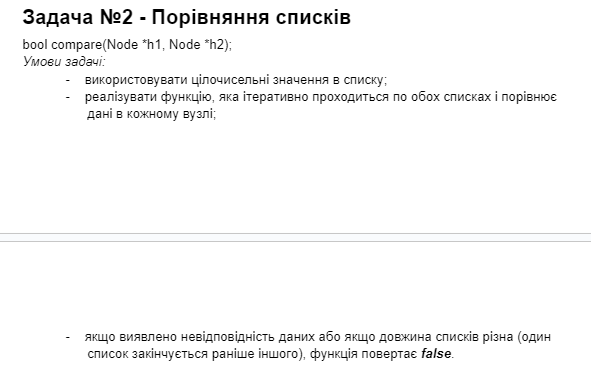
* Варіант завдання: 1
* Деталі завдання:



Завдання № 4. Class Practice Work – Task 1

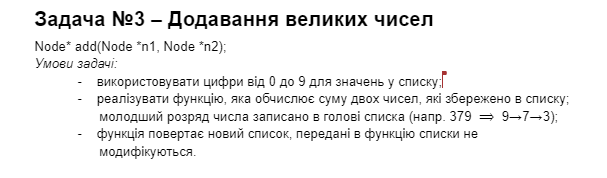
* Деталі завдання:





Завдання № 5. Class Practice Work – Task 2

* Деталі завдання:



Завдання № 6. Self Practice Work – Гра Скринька

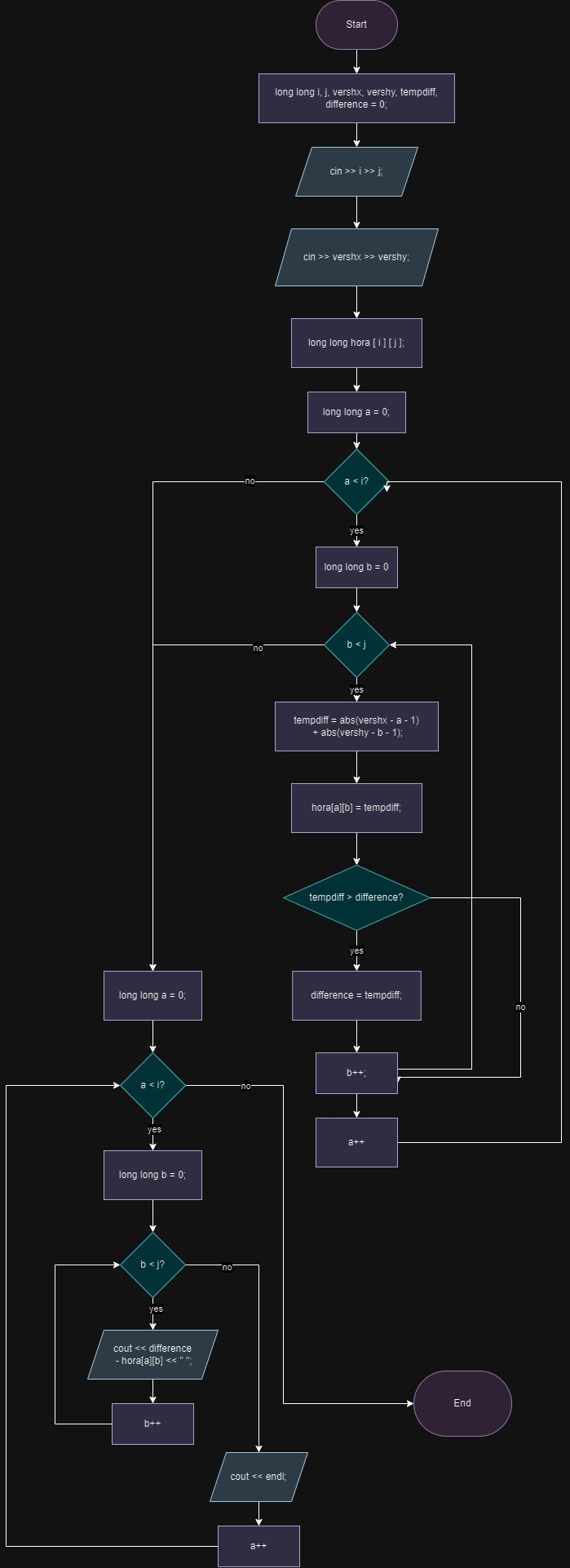
* Деталі завдання:
* Написати гру під назвою Скринька

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

Програма № 1. VNS Lab 10 – Task 1

* Планований час на реалізацію: 3 години

Програма № 2. Algotester Lab 5 – Task 1

* Планований час на реалізацію: 2 години
* Блок-схема:
* 

Програма № 3. Algotester Lab 78 – Task 1

* Планований час на реалізацію: 5 годин

Програма № 4. Class Practice Work – Task 1

* Планований час на реалізацію: 2 години

Програма № 5. Class Practice Work – Task 2

* Планований час на реалізацію: 1 година

Програма № 6. Self Practice Work – Task 1. Гра Скринька

* Планований час на реалізацію: 1 тиждень

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

Додаткова конфігурація середовища непотрібна.

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

**Завдання № 1. VNS Lab 10 – Task 1**

Код до програми:

#include <iostream>

#include <fstream>

#include <cstring>

using **namespace** std;

**struct** Node

{

**char** \*data;

Node \*prev;

Node \*next;

};

**class** DoublyLinkedList

{

**private:**

Node \*head;

Node \*tail;

**public:**

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

Node **\***createNode(**const** **char** **\***value)

{

Node \*newNode = new Node;

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

strcpy(newNode->data, value);

newNode->prev = nullptr;

newNode->next = nullptr;

return newNode;

}

**void** addNode(**const** **char** **\***value)

{

Node \*newNode = createNode(value);

if (!head)

{

head = tail = newNode;

}

else

{

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

}

**void** printList()

{

Node \*current = head;

while (current != nullptr)

{

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

current = current->next;

}

cout << endl;

}

**void** deleteNodeByKey(**const** **char** **\***key)

{

Node \*current = head;

while (current != nullptr)

{

if (strcmp(current->data, key) == 0)

{

if (current == head)

{

head = head->next;

if (head != nullptr)

{

head->prev = nullptr;

}

}

else if (current == tail)

{

tail = tail->prev;

tail->next = nullptr;

}

else

{

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

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

}

delete[] current->data;

delete current;

return;

}

current = current->next;

}

cout << "Елемент з ключем '" << key << "' не знайдений." << endl;

}

**void** addKElemToStartEnd(**int** k)

{

for (**int** i = 0; i < k; ++i)

{

**char** value[100];

cout << "Введіть елемент для додавання на початок та в кінець списку: ";

cin >> value;

addNodeToFront(value);

addNode(value);

}

}

**void** addNodeToFront(**const** **char** **\***value)

{

Node \*newNode = createNode(value);

if (!head)

{

head = tail = newNode;

}

else

{

newNode->next = head;

head->prev = newNode;

head = newNode;

}

}

**void** writeToFile(**const** string **&**filename)

{

ofstream file(filename);

Node \*current = head;

while (current != nullptr)

{

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

current = current->next;

}

file.close();

cout << "Список записано у файл '" << filename << "'" << endl;

}

**void** restoreFromFile(**const** string **&**filename)

{

ifstream file(filename);

if (!file.is\_open())

{

cout << "Не вдалося відкрити файл '" << filename << "'" << endl;

return;

}

**char** value[100];

while (file >> value)

{

addNode(value);

}

file.close();

cout << "Список відновлено з файлу '" << filename << "'" << endl;

}

**void** deleteList()

{

Node \*current = head;

while (current != nullptr)

{

Node \*temp = current;

current = current->next;

delete[] temp->data;

delete temp;

}

head = tail = nullptr;

}

};

**int** main()

{

DoublyLinkedList list;

list.addNode("One");

list.addNode("Two");

list.addNode("Three");

cout << "Друк списку: ";

list.printList();

list.deleteNodeByKey("Two");

**int** K = 2;

list.addKElemToStartEnd(K);

cout << "Оновлений список: ";

list.printList();

list.writeToFile("list.txt");

list.deleteList();

list.restoreFromFile("list.txt");

cout << "Відновлений список: ";

list.printList();

return 0;

}

*Рисунок 2. Код до програми № 1*

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1188>

**Завдання № 2. Algotester Lab 5 – Task 1**

Код до програми:

#include <iostream>

#include <cmath>

using **namespace** std;

**int** main()

{

**long** **long** i, j, vershx, vershy, tempdiff, difference = 0;

cin >> i >> j;

cin >> vershx >> vershy;

**long** **long** hora[i][j];

for (**long** **long** a = 0; a < i; a++)

{

for (**long** **long** b = 0; b < j; b++)

{

tempdiff = abs(vershx - a - 1) + abs(vershy - b - 1);

hora[a][b] = tempdiff;

if (tempdiff > difference)

{

difference = tempdiff;

}

}

}

for (**long** **long** a = 0; a < i; a++)

{

for (**long** **long** b = 0; b < j; b++)

{

cout << difference - hora[a][b] << " ";

}

cout << endl;

}

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1188>

**Завдання № 3. Algotester Lab 78 – Task 1**

Код до програми:

#include <iostream>

using **namespace** std;

**template** <**class** ownType>

**class** Node

{

**public:**

ownType parameter;

Node \*back;

Node \*next;

Node() : parameter(0), back(nullptr), next(nullptr) {}

Node(ownType n, Node **\***prev) : parameter(n), back(prev), next(nullptr) {}

};

**template** <**class** ownType>

**class** List

{

**private:**

Node<ownType> \*head;

**public:**

List() : head(nullptr) {}

**void** insert()

{

**long** **long** index, N;

ownType parameter;

cin >> index >> N;

Node<ownType> \*current = head;

Node<ownType> \*prev = nullptr;

for (**long** **long** i = 0; i < index; i++)

{

prev = current;

current = current->next;

}

for (**long** **long** i = 0; i < N; i++)

{

cin >> parameter;

Node<ownType> \*newNode = new Node<ownType>(parameter, prev);

if (prev)

{

prev->next = newNode;

}

else

{

head = newNode;

}

prev = newNode;

}

if (current)

{

prev->next = current;

current->back = prev;

}

}

**void** erase()

{

**long** **long** index, n;

cin >> index >> n;

Node<ownType> \*prev = nullptr;

Node<ownType> \*current = head;

Node<ownType> \*randomptr;

for (**long** **long** i = 0; i < index; i++)

{

current = current->next;

}

prev = current->back;

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

{

randomptr = current->next;

delete current;

current = randomptr;

}

if (prev)

{

prev->next = current;

}

else

{

head = current;

}

if (current)

{

current->back = prev;

}

}

ownType get() **const**

{

**long** **long** index;

cin >> index;

Node<ownType> \*randomptr = head;

for (**long** **long** i = 0; i < index; i++)

{

randomptr = randomptr->next;

}

return randomptr->parameter;

}

**void** set()

{

**long** **long** index;

ownType parameter;

cin >> index >> parameter;

Node<ownType> \*randomptr = head;

for (**long** **long** i = 0; i < index; i++)

{

randomptr = randomptr->next;

}

randomptr->parameter = parameter;

}

**long** **long** size() **const**

{

**long** **long** length = 0;

Node<ownType> \*randomptr = head;

for (**long** **long** i = 0; randomptr != nullptr; i++)

{

randomptr = randomptr->next;

++length;

}

return length;

}

**void** print() **const**

{

Node<ownType> \*randomptr = head;

for (**long** **long** i = 0; randomptr != nullptr; i++)

{

cout << randomptr->parameter << " ";

randomptr = randomptr->next;

}

cout << endl;

}

};

**int** main()

{

List<**long** **long**> selfcont;

**long** **long** q;

string input;

cin >> q;

for (**long** **long** i = 0; i < q; i++)

{

cin >> input;

if (input == "insert")

{

selfcont.insert();

}

else if (input == "erase")

{

selfcont.erase();

}

else if (input == "size")

{

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

}

else if (input == "get")

{

cout << selfcont.get() << endl;

}

else if (input == "set")

{

selfcont.set();

}

else if (input == "print")

{

selfcont.print();

}

}

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1188>

**Завдання № 4. Class Practice Work – Task 1**

Код до програми:

#include <iostream>

using **namespace** std;

**class** Node

{

**public:**

**const** **char** \*data;

Node \*next;

};

**void** printLinkedList(Node **\***n)

{

while (n != nullptr)

{

cout << n->data << " ";

n = n->next;

}

cout << endl;

}

Node **\***reverseLinkedList(Node **\***head)

{

Node \*back = nullptr;

Node \*current = head;

Node \*next = nullptr;

while (current != nullptr)

{

next = current->next;

current->next = back;

back = current;

current = next;

}

return back;

}

**bool** compareLinkedLists(Node **\***L1, Node **\***L2)

{

while (L1 != nullptr && L2 != nullptr)

{

if (L1->data != L2->data)

{

return false;

}

L1 = L1->next;

L2 = L2->next;

}

return (L1 == nullptr && L2 == nullptr);

}

**int** main()

{

Node \*head1 = nullptr;

Node \*second1 = nullptr;

Node \*third1 = nullptr;

head1 = new Node();

second1 = new Node();

third1 = new Node();

head1->data = "Чайник";

head1->next = second1;

second1->data = "Стілець";

second1->next = third1;

third1->data = "Ліжко";

third1->next = nullptr;

cout << "Original Linked List 1: ";

printLinkedList(head1);

Node \*head2 = nullptr;

Node \*second2 = nullptr;

Node \*third2 = nullptr;

head2 = new Node();

second2 = new Node();

third2 = new Node();

head2->data = "Чайник";

head2->next = second2;

second2->data = "Стілець";

second2->next = third2;

third2->data = "Ліжко";

third2->next = nullptr;

cout << "Original Linked List: ";

printLinkedList(head2);

head1 = reverseLinkedList(head1);

cout << "Reversed Linked List: ";

printLinkedList(head1);

**bool** areIdentical = compareLinkedLists(head1, head2);

cout << "Are the linked lists identical? " << (areIdentical ? "Yes" : "No") << endl;

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1188>

**Завдання № 5. Class Practice Work – Task 2**

Код до програми:

#include <iostream>

using **namespace** std;

**class** Node

{

**public:**

**int** data;

Node \*next;

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

};

**void** printLinkedList(Node **\***n)

{

while (n != nullptr)

{

cout << n->data;

if (n->next != nullptr)

{

cout << " -> ";

}

n = n->next;

}

cout << endl;

}

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

{

Node \*Head\_r = nullptr;

Node \*Tail\_r = 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 %= 10;

Node \*newNode = new Node(sum);

if (Head\_r == nullptr)

{

Head\_r = newNode;

Tail\_r = Head\_r;

}

else

{

Tail\_r->next = newNode;

Tail\_r = newNode;

}

}

return Head\_r;

}

**int** main()

{

Node \*num = new Node(9);

num->next = new Node(7);

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

Node \*num2 = new Node(2);

num2->next = new Node(8);

cout << "Number 1: ";

printLinkedList(num);

cout << "Number 2: ";

printLinkedList(num2);

Node \*sumResult = add(num, num2);

cout << "Sum: ";

printLinkedList(sumResult);

return 0;

}

**Завдання № 6. Class Practice Work – Task 2**

Код до програми:

#include <iostream>

#include <stack>

using **namespace** std;

**struct** TreeNode

{

**int** value;

TreeNode \*left;

TreeNode \*right;

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

};

TreeNode **\***mirrorflip(TreeNode **\***root)

{

if (root == nullptr)

{

return nullptr;

}

TreeNode \*mirroredRoot = new TreeNode(root->value);

mirroredRoot->left = mirrorflip(root->right);

mirroredRoot->right = mirrorflip(root->left);

return mirroredRoot;

}

**void** tree\_sum(TreeNode **\***root)

{

if (!root)

{

return;

}

stack<TreeNode \*> nodeStack;

TreeNode \*current = root;

TreeNode \*previous = nullptr;

while (current || !nodeStack.empty())

{

while (current)

{

nodeStack.push(current);

current = current->left;

}

current = nodeStack.top();

if (current->right && current->right != previous)

{

current = current->right;

}

else

{

**int** val = current->value;

if (current->left)

{

val += current->left->value;

}

if (current->right)

{

val += current->right->value;

}

current->value = val;

nodeStack.pop();

previous = current;

current = nullptr;

}

}

}

**void** printTree(**const** TreeNode **\***root)

{

if (root != nullptr)

{

printTree(root->left);

cout << root->value << " ";

printTree(root->right);

}

}

**int** main()

{

TreeNode \*tree = new TreeNode(2);

tree->left = new TreeNode(5);

tree->right = new TreeNode(8);

tree->left->left = new TreeNode(11);

tree->left->right = new TreeNode(14);

cout << "Origin tree: ";

printTree(tree);

cout << endl;

TreeNode \*mirroredTree = mirrorflip(tree);

cout << "Mirrored tree: ";

printTree(mirroredTree);

cout << endl;

tree\_sum(tree);

cout << "Tree sum:" << endl;

printTree(tree);

return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1188>

**Завдання № 7. Self Practice Work – Task 1. Скринька**

Код до програми:

#include <iostream>

#include <algorithm>

#include <random>

#include <vector>

#include <ctime>

#include <limits>

#include <cmath>

using **namespace** std;

**int** main()

{

vector<**char**> arr = {'6', '6', '6', '6', '7', '7', '7', '7', '8', '8', '8', '8', '9', '9', '9', '9', '1', '1', '1', '1', 'J', 'J', 'J', 'J', 'Q', 'Q', 'Q', 'Q', 'K', 'K', 'K', 'K', 'A', 'A', 'A', 'A'};

random\_device rd;

mt19937 g(rd());

shuffle(arr.begin(), arr.end(), g);

vector<**char**> newArr(arr.begin(), arr.begin() + 4);

arr.erase(arr.begin(), arr.begin() + 4);

vector<**char**> newArrr(arr.begin(), arr.begin() + 4);

arr.erase(arr.begin(), arr.begin() + 4);

**bool** svinguess = false;

**bool** hoholguess = true;

**bool** found = false;

**bool** found2 = false;

**bool** hohol = false;

**bool** foundFour1 = false;

**bool** foundFour2 = false;

**bool** randombool = false;

**char** temp;

**char** temp1;

**char** temp2;

**char** temp3;

**char** n;

**char** random;

**int** foundfourplayer = 0;

**int** foundfoursvin = 0;

**int** randomIndex;

string name;

string answer;

cout << "Придуймайте назву для вашого гравця: " << endl;

cin >> name;

cout << "Привіт, " << name << ", вас вітає гра 'Скринька', бажаєте ознайомитись з правилами гри чи пропустите цей етап? (Yes - ознайомлення з правилами, No - старт гри): ";

cin >> answer;

if (answer == "Yes" || answer == "yes")

{

goto rules;

}

if (answer == "No" || answer == "no")

{

goto playercards;

}

rules:

cout << "\n--------------------------------------------------------------------------------------------------------------------------------" << endl;

cout << "Кожен гравець отримує по 4 карти. Гра починається з питання вашому противнику чи є у нього карта певного номіналу, наприклад: \n - У вас є валет? \n - Так \nГравець забирає всі вальти і знову задає питання. Коли Гравець не вгадує карту, він тягне карту з колоди та хід переходить противнику. Якщо хтось з гравців збирає 4 карти одного й того самого номіналу він відкладає їх зі своєї руки та відкладає в сторону тим самим отримує +1 очко. Задача кожного з гравців набрати більше очків ніж противник. Можна питати лише свої карти. Якщо в когось з гравців закінчується карти, гра закінчиться та виводиться рахунок. Успіхів!" << endl;

cout << "--------------------------------------------------------------------------------------------------------------------------------\n"

<< endl;

playercards:

hohol = false;

found = false;

foundFour1 = false;

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Карти " << name << ": ";

for (**char** card : newArr)

{

cout << card << " ";

}

cout << endl;

cout << "---------------------------------------" << endl;

cout << name << ", питай карту, яку хочеш забрати: " << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " << endl;

cin >> n;

for (**char** card : newArr)

{

if (n == card)

{

found = true;

goto hoholguess;

}

}

if (found == false)

{

cout << "Ти кого вирішив обманути, в тебе нема такої карти! " << endl;

goto playercards;

}

hoholguess:

if (hoholguess == true)

{

if (newArr.size() == 0)

{

temp3 = arr[0];

arr.erase(next(arr.begin(), 0));

newArr.push\_back(temp3);

}

for (**int** i = 0; i < newArrr.size(); i++)

{

if (n == newArrr[i])

{

hohol = true;

temp2 = newArrr[i];

newArrr.erase(next(newArrr.begin(), i));

newArr.push\_back(temp2);

i--;

}

if (i == (newArrr.size() - 1) && hohol == true)

{

cout << "\n---------------------------------------" << endl;

cout << name << ", ти сьогодні фартовий, забираєш всі: " << temp2 << " в КАБАНОСА" << endl;

cout << "--------------------------------------- " << endl;

foundFour1 = true;

goto foundFour;

}

}

temp3 = arr[0];

arr.erase(next(arr.begin(), 0));

newArr.push\_back(temp3);

cout << "\n===========================================================" << endl;

cout << "Нажаль ти не вгадав, тепер черга Кабаноса вгадувати карту!" << endl;

cout << "===========================================================" << endl;

svinguess = true;

goto foundFour;

}

svinguess:

if (svinguess == true)

{

if (newArrr.size() == 0)

{

temp3 = arr[0];

arr.erase(next(arr.begin(), 0));

newArrr.push\_back(temp3);

}

svinguess = false;

foundFour2 = false;

random:

srand(static\_cast<**unsigned** **int**>(time(nullptr)));

**char** randomIndex = rand() % newArrr.size();

if (randombool == true)

{

if (random == newArrr[randomIndex])

{

goto random;

}

}

random = newArrr[randomIndex];

randombool = false;

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Кабанос хоче вкрасти в тебе: " << newArrr[randomIndex] << endl;

cout << "---------------------------------------" << endl;

for (**int** i = 0; i < newArr.size(); i++)

{

if (newArrr[randomIndex] == newArr[i])

{

temp = newArr[i];

newArr.erase(next(newArr.begin(), i));

newArrr.push\_back(temp);

found2 = true;

i--;

}

}

if (found2 == true)

{

cout << "Кабанос вгадав карту: " << newArrr[randomIndex] << " та забариє всіх в " << name << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

svinguess = true;

found2 = false;

foundFour2 = true;

randombool = true;

goto foundFour4;

}

else

{

temp1 = arr[0];

arr.erase(next(arr.begin(), 0));

newArrr.push\_back(temp1);

cout << "Кабанос не вгадав карту, настала твоя черга! " << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

svinguess = false;

foundFour2 = false;

randombool = false;

goto foundFour4;

}

}

foundFour:

std::sort(newArr.begin(), newArr.end());

if (newArr.size() > 3)

{

for (**int** i = 0; i <= newArr.size() - 4; ++i)

{

if (newArr[i] == newArr[i + 1] && newArr[i] == newArr[i + 2] && newArr[i] == newArr[i + 3])

{

cout << "\n=======================================" << endl;

cout << name << " зібрав всі: " << newArr[i] << endl;

cout << "=======================================" << endl;

newArr.erase(newArr.begin() + i, newArr.begin() + i + 4);

foundfourplayer++;

break;

}

}

}

if (arr.size() == 0 && newArr.size() == 0 && newArrr.size() == 0)

{

if (foundfourplayer > foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << name << "виграв Кабаноса, тепер він відправляється на сало! Рахунок: " << foundfourplayer << ":" << foundfoursvin << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (arr.size() == 0 && newArr.size() == 0 && newArrr.size() == 0)

{

if (foundfourplayer < foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Кабанос виграв, " << name << " йде засмучений додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (arr.size() == 0 && newArr.size() == 0 && newArrr.size() == 0)

{

if (foundfourplayer == foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Перемогла дружба, Кабанос і " << name << " йдуть додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (newArrr.size() == 0)

{

cout << "\n+++++++++++++++++++++++++++++++" << endl;

cout << "В Кабаноса закінчились карти. Підводимо підсумки! " << endl;

cout << "+++++++++++++++++++++++++++++++" << endl;

if (foundfourplayer > foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << name << "виграв Кабаноса, тепер він відправляється на сало! Рахунок: " << foundfourplayer << ":" << foundfoursvin << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer < foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Кабанос виграв, " << name << " йде засмучений додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer == foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Перемогла дружба, Кабанос і " << name << " йдуть додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (newArr.size() == 0)

{

cout << "\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

cout << "У " << name << " закінчились карти. Підводимо підсумки гри! " << endl;

cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

if (foundfourplayer > foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << name << " виграв Кабаноса, тепер він відправляється на сало! Рахунок: " << foundfourplayer << ":" << foundfoursvin << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer < foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Кабанос виграв, " << name << " йде засмучений додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer == foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Перемогла дружба, Кабанос і " << name << " йдуть додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

std::cin.clear();

if (foundFour1 == true)

{

goto playercards;

}

if (foundFour1 == false)

{

goto svinguess;

}

foundFour4:

std::sort(newArrr.begin(), newArrr.end());

if (newArrr.size() > 3)

{

for (**int** i = 0; i <= newArrr.size() - 4; ++i)

{

if (newArrr[i] == newArrr[i + 1] && newArrr[i] == newArrr[i + 2] && newArrr[i] == newArrr[i + 3])

{

cout << "\n=======================================" << endl;

cout << "Кабанос зібрав всі: " << newArrr[i] << " та відкладає їх в сторону" << endl;

cout << "=======================================" << endl;

newArrr.erase(newArrr.begin() + i, newArrr.begin() + i + 4);

foundfoursvin++;

break;

}

}

}

if (arr.size() == 0 && newArr.size() == 0 && newArrr.size() == 0)

{

if (foundfourplayer > foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << name << "виграв Кабаноса, тепер він відправляється на сало! Рахунок: " << foundfourplayer << ":" << foundfoursvin << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (arr.size() == 0 && newArr.size() == 0 && newArrr.size() == 0)

{

if (foundfourplayer < foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Кабанос виграв, " << name << " йде засмучений додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (arr.size() == 0 && newArr.size() == 0 && newArrr.size() == 0)

{

if (foundfourplayer == foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Перемогла дружба, Кабанос і " << name << " йдуть додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (newArrr.size() == 0)

{

cout << "\n+++++++++++++++++++++++++++++++" << endl;

cout << "В Кабаноса закінчились карти. Підводимо підсумки! " << endl;

cout << "+++++++++++++++++++++++++++++++" << endl;

if (foundfourplayer > foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << name << " виграв Кабаноса, тепер він відправляється на сало! Рахунок: " << foundfourplayer << ":" << foundfoursvin << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer < foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Кабанос виграв, " << name << " йде засмучений додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer == foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Перемогла дружба, Кабанос і " << name << " йдуть додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

else if (newArr.size() == 0)

{

cout << "\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

cout << "У " << name << " закінчились карти. Підводимо підсумки гри! " << endl;

cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;

if (foundfourplayer > foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << name << "виграв Кабаноса, тепер він відправляється на сало! Рахунок: " << foundfourplayer << ":" << foundfoursvin << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer < foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Кабанос виграв, " << name << " йде засмучений додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

else if (foundfourplayer == foundfoursvin)

{

cout << "\nWIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

cout << "Перемогла дружба, Кабанос і " << name << " йдуть додому, рахунок: " << foundfoursvin << ":" << foundfourplayer << endl;

cout << "WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN WIN" << endl;

return 0;

}

}

std::cin.clear();

if (foundFour2 == true)

{

goto svinguess;

}

if (foundFour2 == false)

{

goto playercards;

}

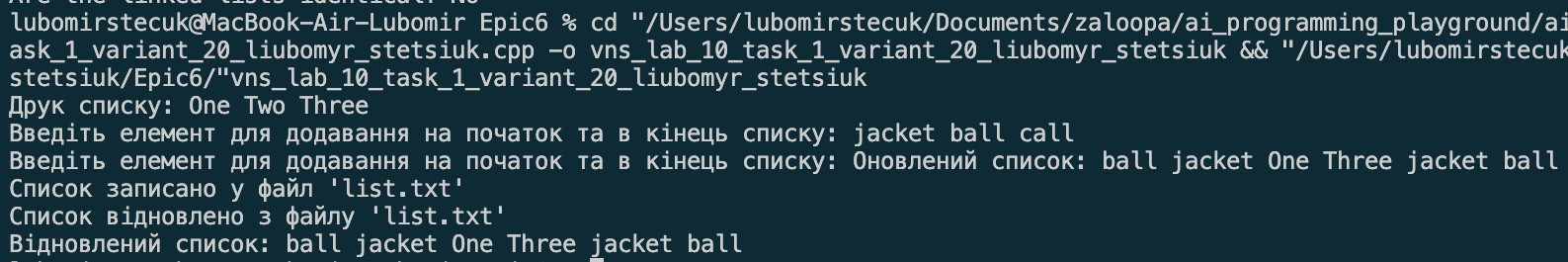
cout << "hohol" << endl;

return 0;

}

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

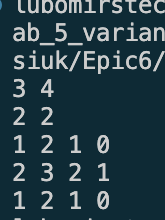
**Завдання № 1. VNS Lab 10 – Task 1**

****

*Рисунок 3. Результат виконання до програми № 1*

Час затрачений на виконання завдання: 3 години

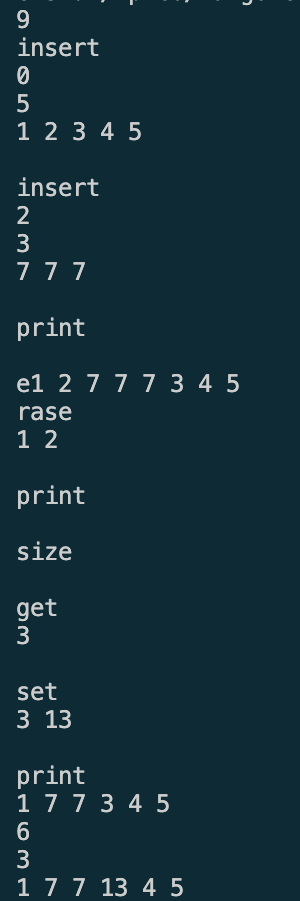
**Завдання № 2. Algotester Lab 5 – Task 1**



*Рисунок 4. Результат виконання до програми № 2*

Час затрачений на виконання завдання: 2 години

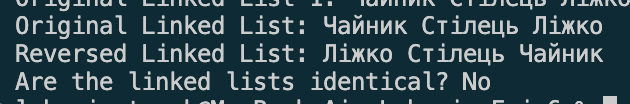
**Завдання № 3. Algotester Lab 78 – Task 1**

****

*Рисунок 5. Результат виконання до програми № 3*

Час затрачений на виконання завдання: 5 годин

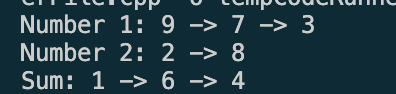
**Завдання № 4. Class Practice Work – Task 1**

****

*Рисунок 6. Результат виконання до програми № 4*

Час затрачений на виконання завдання: 2 години

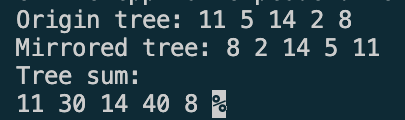
**Завдання № 5. Class Practice Work – Task 2**

****

*Рисунок 7. Результат виконання до програми № 5*

Час затрачений на виконання завдання: 1 година

**Завдання № 6. Class Practice Work – Task 3**



*Рисунок 8. Результат виконання до програми № 5*

Час затрачений на виконання завдання: 1 година

**Завдання № 7. Self Practice Work – Скринька**

Придуймайте назву для вашого гравця:

Любомир

Привіт, Любомир, вас вітає гра 'Скринька', бажаєте ознайомитись з правилами гри чи пропустите цей етап? (Yes - ознайомлення з правилами, No - старт гри): Yes

--------------------------------------------------------------------------------------------------------------------------------

Кожен гравець отримує по 4 карти. Гра починається з питання вашому противнику чи є у нього карта певного номіналу, наприклад:

- У вас є валет?

- Так

Гравець забирає всі вальти і знову задає питання. Коли Гравець не вгадує карту, він тягне карту з колоди та хід переходить противнику. Якщо хтось з гравців збирає 4 карти одного й того самого номіналу він відкладає їх зі своєї руки та відкладає в сторону тим самим отримує +1 очко. Задача кожного з гравців набрати більше очків ніж противник. Можна питати лише свої карти. Якщо в когось з гравців закінчується карти, гра закінчиться та виводиться рахунок. Успіхів!

--------------------------------------------------------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 J Q 7

---------------------------------------

Любомир, питай карту, яку хочеш забрати:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

J

===========================================================

Нажаль ти не вгадав, тепер черга Кабаноса вгадувати карту!

===========================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Кабанос хоче вкрасти в тебе: 8

---------------------------------------

Кабанос не вгадав карту, настала твоя черга!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 7 J Q Q

---------------------------------------

Любомир, питай карту, яку хочеш забрати:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Q

===========================================================

Нажаль ти не вгадав, тепер черга Кабаноса вгадувати карту!

===========================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Кабанос хоче вкрасти в тебе: A

---------------------------------------

Кабанос вгадав карту: A та забариє всіх в Любомир

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Кабанос хоче вкрасти в тебе: 8

---------------------------------------

Кабанос не вгадав карту, настала твоя черга!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 7 J Q Q

---------------------------------------

Любомир, питай карту, яку хочеш забрати:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7

===========================================================

Нажаль ти не вгадав, тепер черга Кабаноса вгадувати карту!

===========================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Кабанос хоче вкрасти в тебе: A

---------------------------------------

Кабанос не вгадав карту, настала твоя черга!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 1 7 J Q Q

---------------------------------------

Любомир, питай карту, яку хочеш забрати:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

---------------------------------------

Любомир, ти сьогодні фартовий, забираєш всі: 1 в КАБАНОСА

---------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 1 1 7 J Q Q

---------------------------------------

Любомир, питай карту, яку хочеш забрати:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Q

---------------------------------------

Любомир, ти сьогодні фартовий, забираєш всі: Q в КАБАНОСА

---------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 1 1 7 J Q Q Q

---------------------------------------

Любомир, питай карту, яку хочеш забрати:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

J

===========================================================

Нажаль ти не вгадав, тепер черга Кабаноса вгадувати карту!

===========================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Кабанос хоче вкрасти в тебе: 6

---------------------------------------

Кабанос не вгадав карту, настала твоя черга!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Карти Любомир: 1 1 1 7 J J Q Q Q

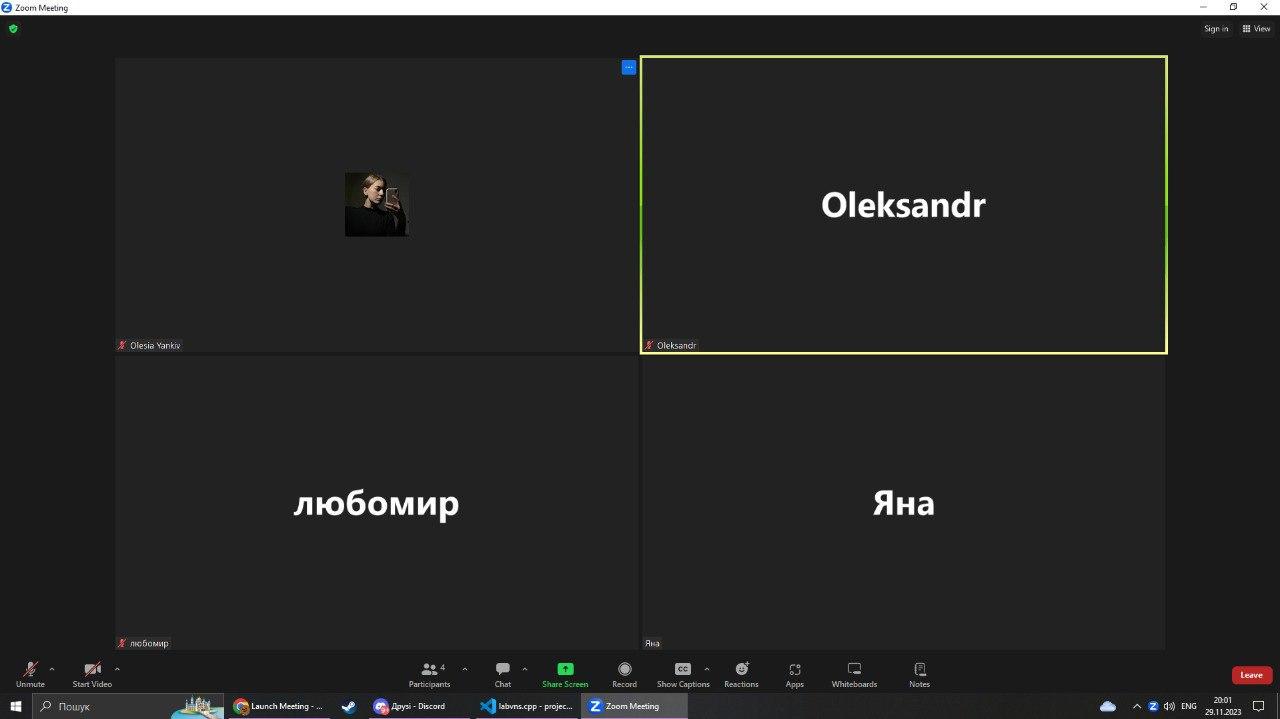
---------------------------------------

Любомир, питай карту, яку хочеш забрати:

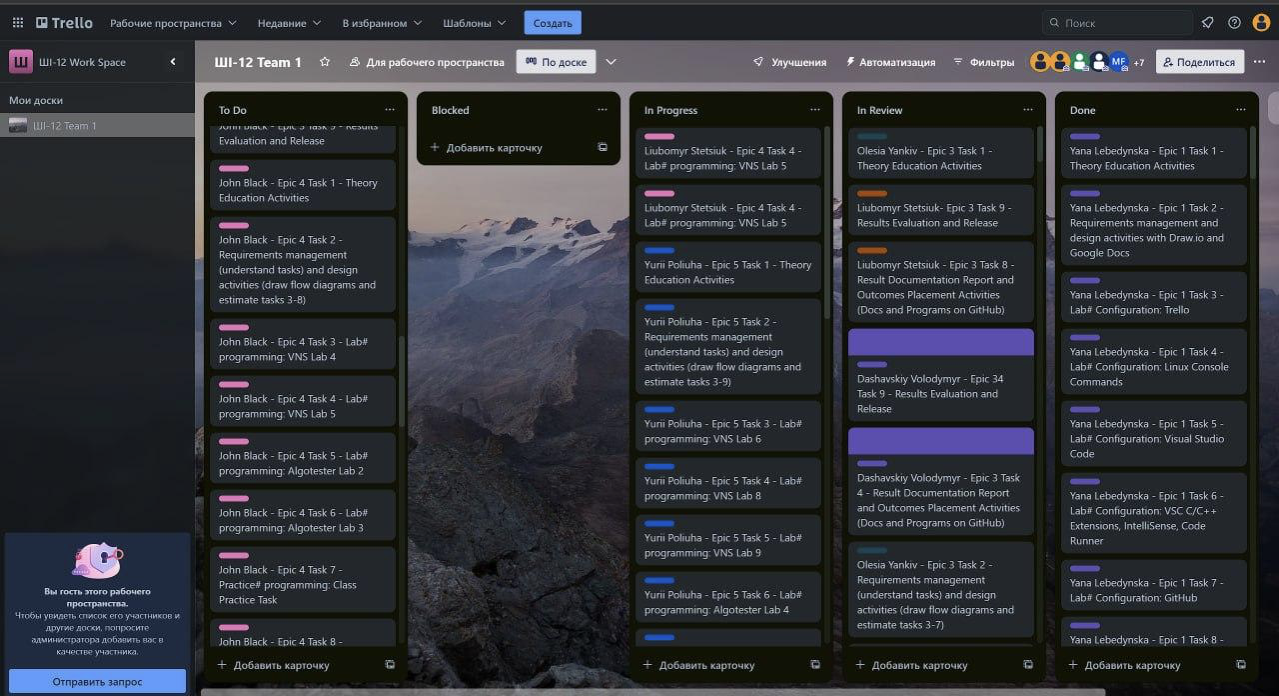
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Час затрачений на виконання завдання: 1 година

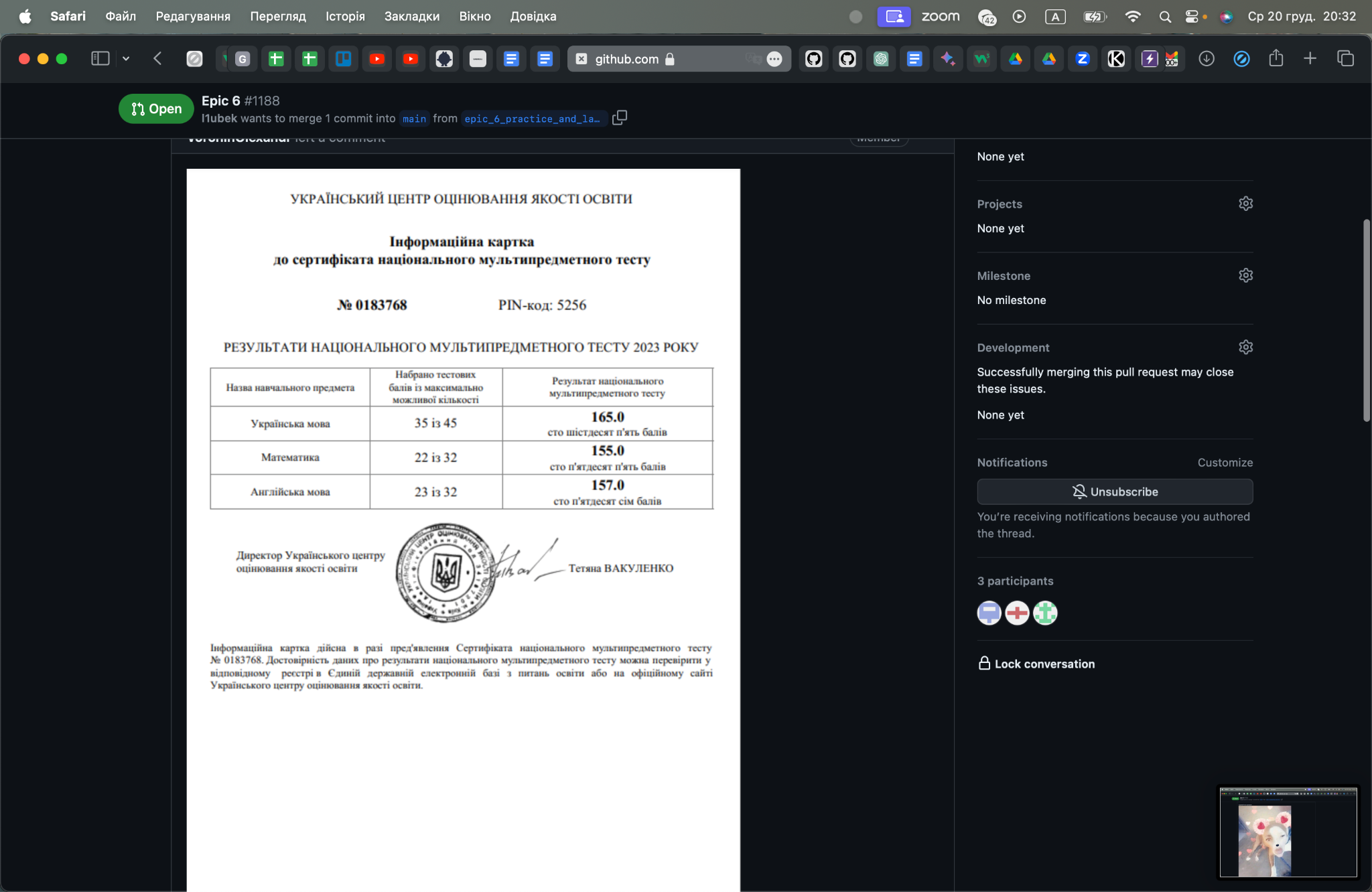
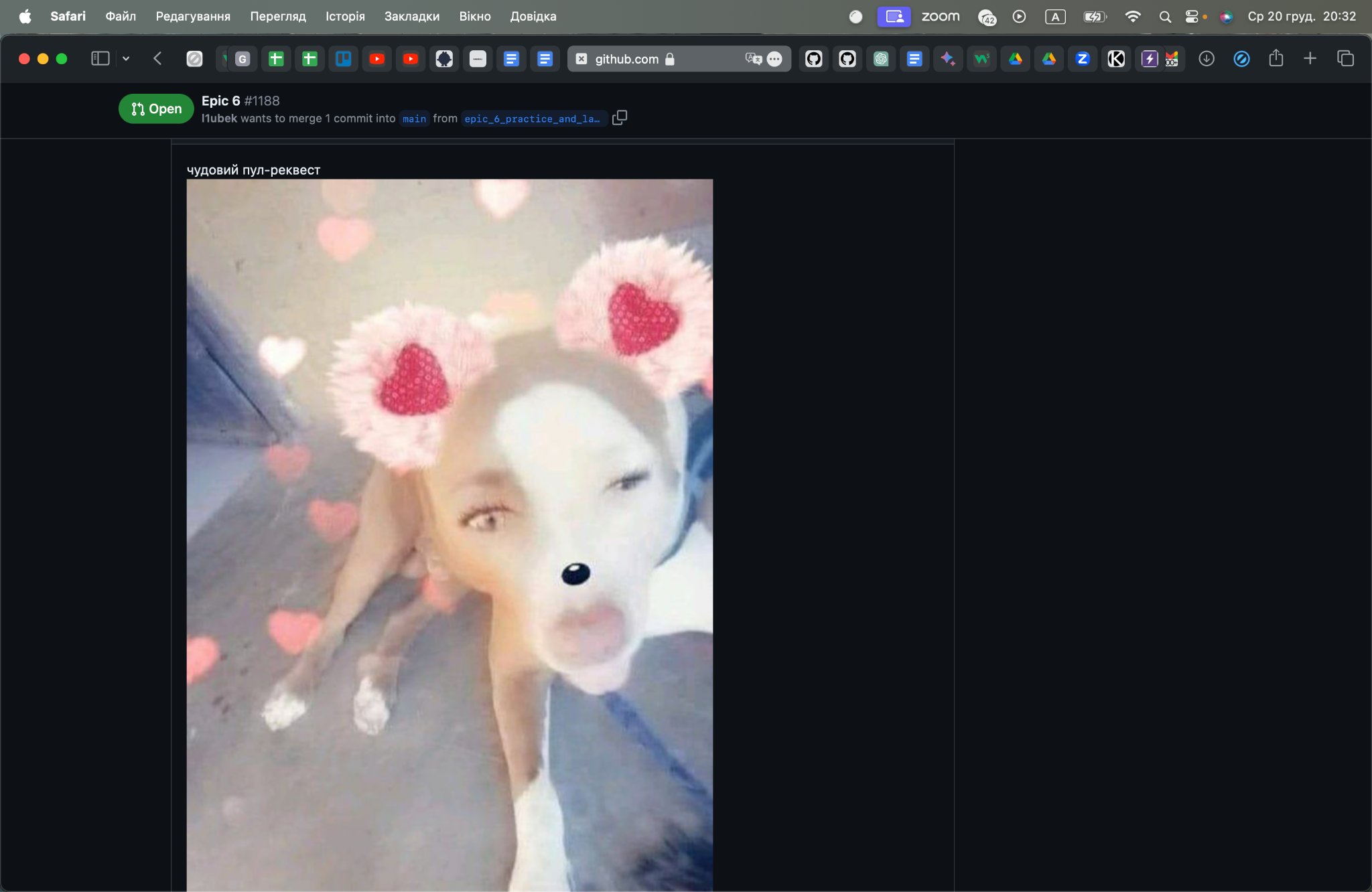
**6. Кооперація з командою:**



*Рисунок 8. Зустріч з командою для обговорення питань по завданнях*



*Рисунок 9. Прогрес в Trello*

*Рисунок 10. Коментарі до пул-реквесту*

**Висновки:**

Виконуючи дані практичні і лабораторні роботи, я ознайомився з основними динамічними структурами даних, такими як стеки, черги, списки і дерева, зрозумів їхні особливості, роботу та функції. Вивчення операцій над цими структурами включило в себе розуміння способів додавання, вилучення та обробки даних, що зберігаються у цих структурах. Практичні навички реалізації алгоритмів з використанням динамічних структур даних в мові програмування C++ допомогли закріпити ці знання та вміння у практичному застосуванні.