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: «Основи програмування»

до:

ВНС Лабораторної Роботи № 10
Алготестер Лабораторної Роботи № 5
Алготестер Лабораторної Роботи № 7-8
Практичних Робіт до блоку № 6

Виконала:

Студентка групи ШІ-13
Паничевська Ярина Ернестівна

Львів 2024

Тема:

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

Мета: Розібрати такі теми, як :

Основи Динамічних Структур Даних:

- Вступ до динамічних структур даних: визначення та важливість
- Виділення пам'яті для структур даних (**stack** і **heap**)
- Приклади простих динамічних структур: динамічний масив

Стек:

- Визначення та властивості стеку
- Операції **push**, **pop**, **top**: реалізація та використання
- Приклади використання стеку: обернений польський запис, перевірка балансу дужок
- Переповнення стеку

Черга:

- Визначення та властивості черги
- Операції **enqueue**, **dequeue**, **front**: реалізація та застосування
- Приклади використання черги: обробка подій, алгоритми планування
- Розширення функціоналу черги: пріоритетні черги

Зв'язні Списки:

- Визначення однозв'язного та двозв'язного списку
- Принципи створення нових вузлів, вставка між існуючими, видалення, створення кільця(**circular linked list**)
- Основні операції: обхід списку, пошук, доступ до елементів, об'єднання списків
- Приклади використання списків: управління пам'яттю, FIFO та LIFO структури

Дерева:

- Вступ до структури даних "дерево": визначення, типи
- Бінарні дерева: вставка, пошук, видалення
- Обхід дерева: **preorder**, **inorder**, **postorder**
- Застосування дерев: дерева рішень, хеш-таблиці
- Складніші приклади дерев: AVL, Червоно-чорне дерево

Алгоритми Обробки Динамічних Структур:

- Основи алгоритмічних патернів: ітеративні, рекурсивні
- Алгоритми пошуку, сортування даних, додавання та видалення елементів

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Завдання № 1 (Class practice work)

Умова до пункту 1:

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

- використовувати цілочисельні значення в списку;
- реалізувати метод реверсу;
- реалізувати допоміжний метод виведення вхідного і обернутого списків;

Умова до пункту 2:

Реалізувати порівняння двох списків: bool compare(Node *h1, Node *h2);

- використовувати цілочисельні значення в списку;
- реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;
- якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

Умова до пункту 3 :

Реалізувати додавання двох чисел у вигляді списків: Node* add(Node *n1, Node *n2);

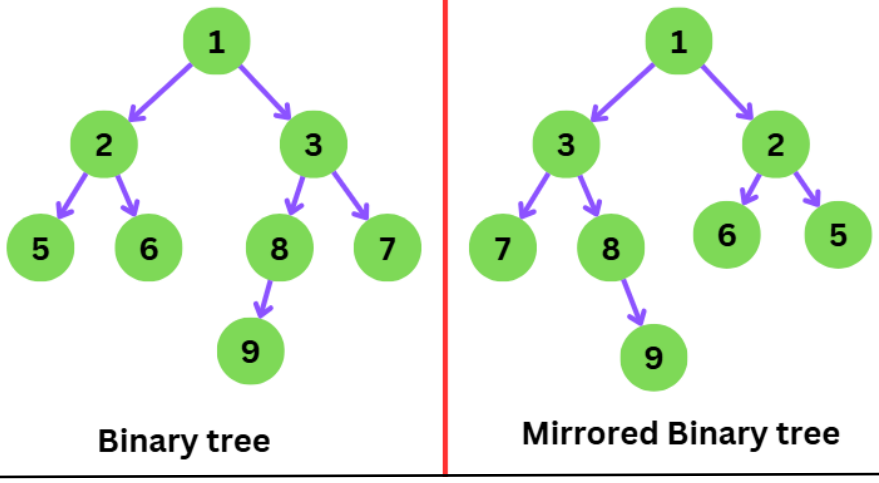
- використовувати цифри від 0 до 9 для значень у списку;
- реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379 $\Longrightarrow$ 9→7→3);
- функція повертає новий список, передані в функцію списки не модифікуються.

Умова до пункту 4 :

Реалізувати відзеркалення дерева: TreeNode *create_mirror_flip(TreeNode *root);

- використовувати цілі числа для значень у вузлах дерева;
- реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева;
- функція повертає нове дерево, передане в функцію дерево не модифікується.

Приклад:

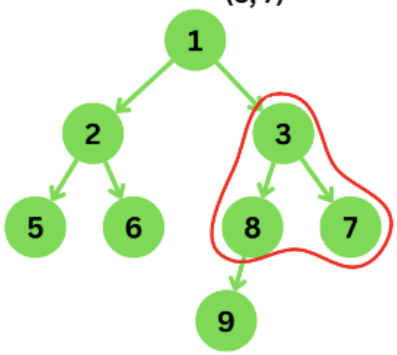


Умова до пункту 5 :

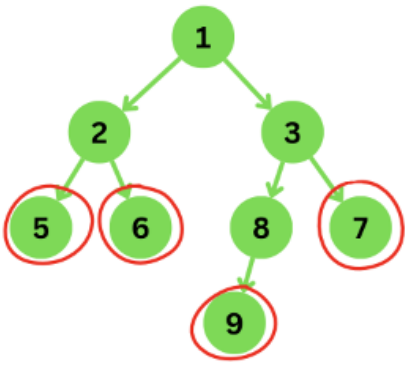
Реалізувати нове дерево в якому кожна батьківська вершина – це сума двох його дочірніх вузлів: void tree_sum(TreeNode *root);

- використовувати цілочисельні значення у вузлах дерева;
- реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів;
- вузол-листок не змінює значення;
- значення змінюються від листків до кореня дерева.

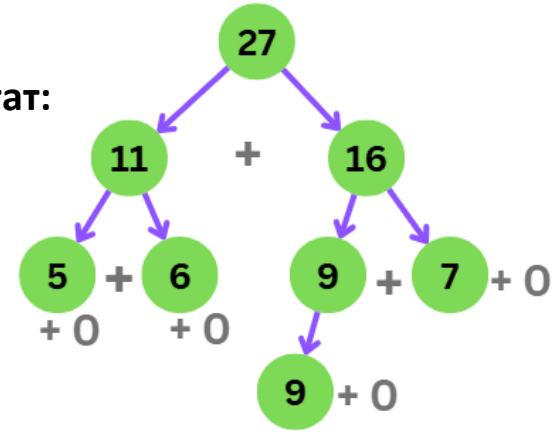
Дочірні вузли батьківського вузла 3
(8, 7)



Дочірні вузли батьківських вузлів 5, 6, 7, 9
(0)



Кінцевий результат:



Завдання № 2 (VNS Lab10. V 22)

Вимога: Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом.

Для кожного варіанту розробити такі функції:

1. Створення списку.

2. Додавання елемента в список (у відповідності зі своїм варіантом).

3. Знищення елемента зі списку (у відповідності зі своїм варіантом).

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

Умова: Записи в лінійному списку містять ключове поле типу string.

Сформувати двонаправлений список. Знищити елемент із заданим

ключем. Додати K елементів перед елементом із заданим ключем.

Завдання № 3 (Algotester Lab5. V 3)

Умова:

У вас є карта гори розміром N × M.
Також ви знаєте координати {x,y} , у яких знаходиться вершина гори.
Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.

Клітинки, які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

Input:

У першому рядку 2 числа N та M - розміри карти
у другому рядку 2 числа x та y - координати піку гори

Output:

N рядків по M елементів в рядку через пробіл - висоти карти.

Завдання № 4 (Algotester Lab7-8. V 3)

Умова: Ваше завдання - власноруч реалізувати структуру даних "Двійкове дерево пошуку".
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його параметри.

Input:

Ціле число Q - кількість запитів.
У наступних рядках Q запитів у зазначеному в умові форматі.

Output:

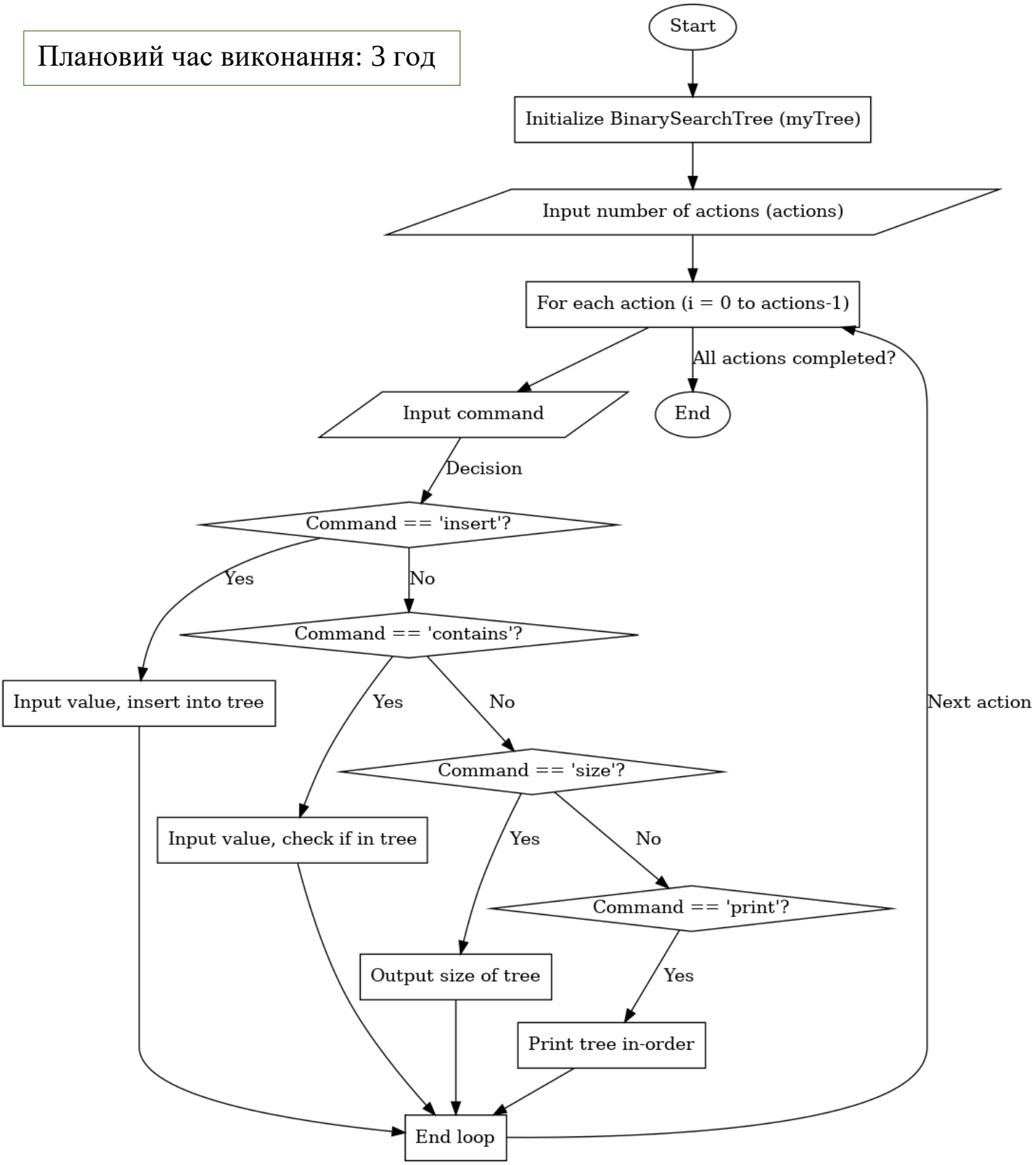
Відповіді на запити у зазначеному в умові форматі.

- **Вставка**:
  Ідентифікатор - **insert**
  Ви отримуєте ціле число **value** - число, яке треба вставити в дерево.
- **Пошук**:
  Ідентифікатор - **contains**
  Ви отримуєте ціле число **value** - число, наявність якого у дереві необхідно перевірити.
  Якщо **value** наявне в дереві - ви виводите **Yes**, у іншому випадку **No**.
- **Визначення розміру**:
  Ідентифікатор - **size**
  Ви не отримуєте аргументів.
  Ви виводите кількість елементів у дереві.
- **Вивід дерева на екран**
  Ідентифікатор - **print**
  Ви не отримуєте аргументів.
  Ви виводите усі елементи дерева через пробіл.
  Реалізувати використовуючи перегрузку оператора <<

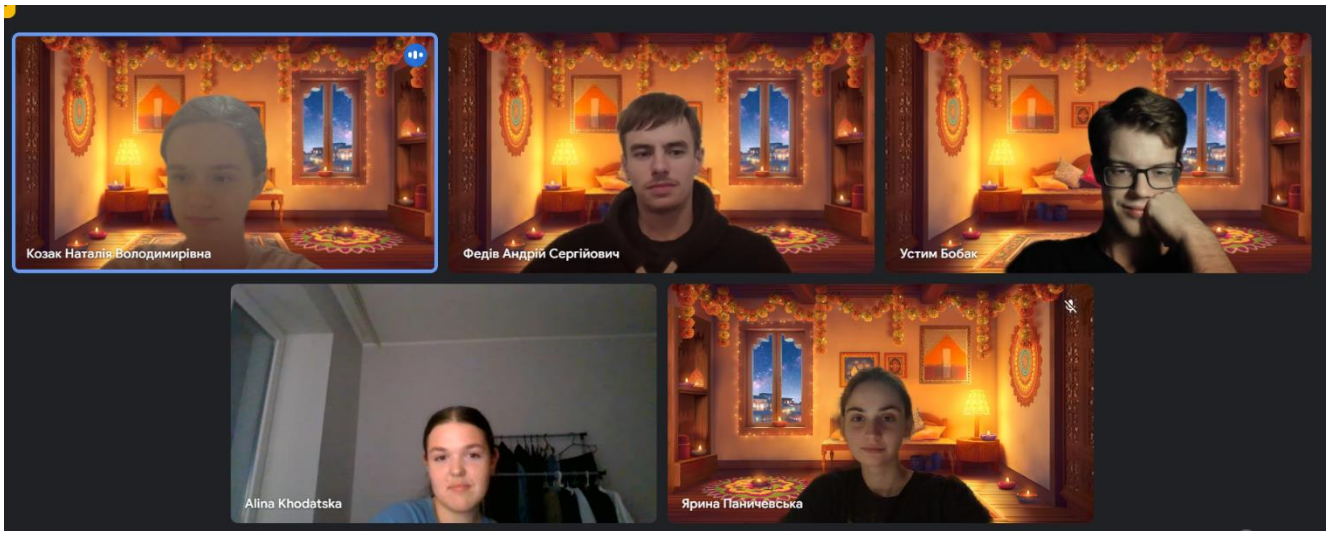
2. *Дизайн та планована оцінка часу виконання завдань:*

Завдання № 4 (Algotester Lab7-8. V 3)

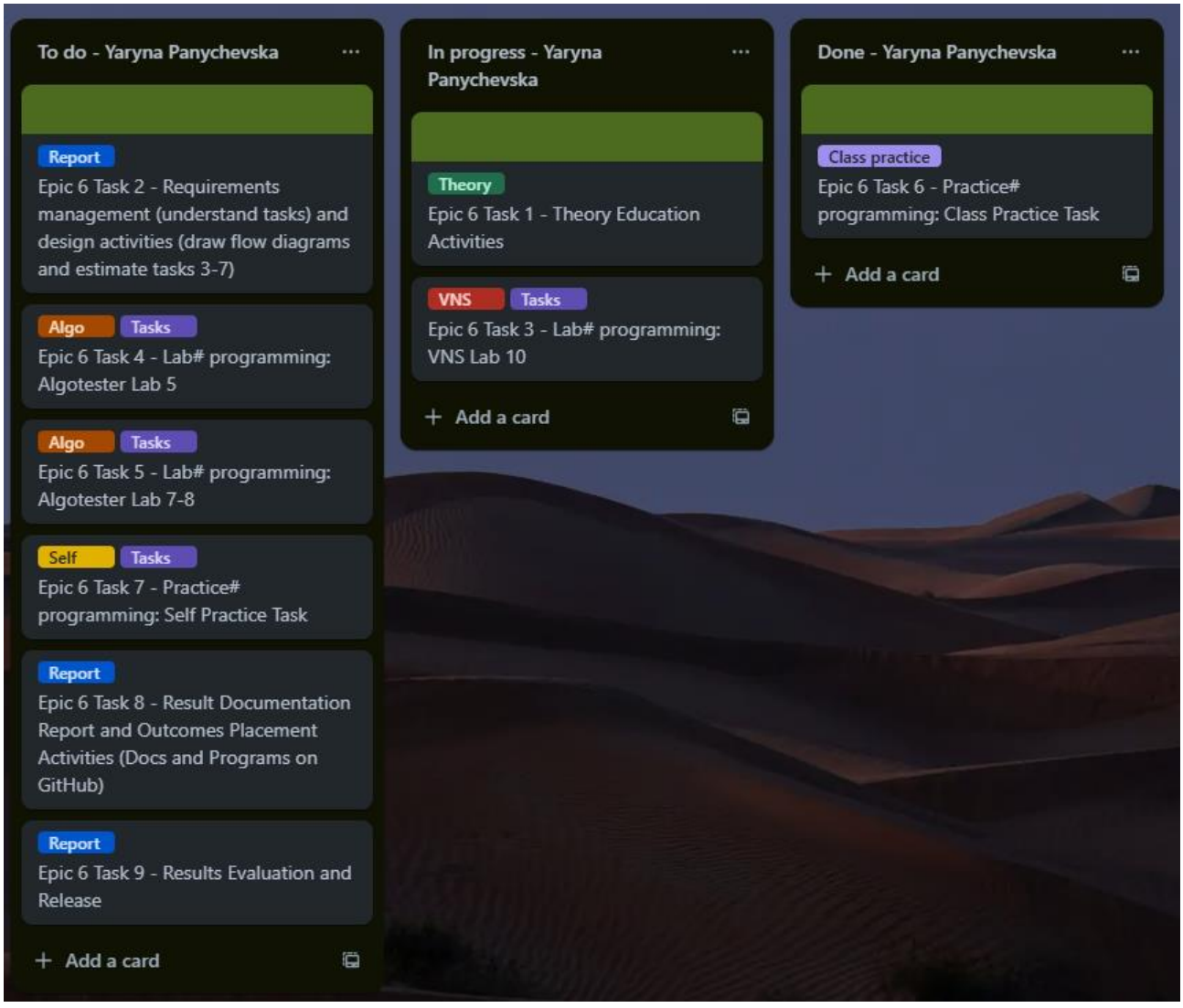
Плановий час виконання: 3 год



3. *Дошка, зустріч з командою:*



Онлайн-зустріч з командою



Дошка в Trello

4. *Результати виконання завдань, тестування:*

<u>Завдання № 1 (Class practice work)</u> | Затрачений час – 5-6 год

Пункт 1:

```
Enter the number of elements: 3
Enter 3 elements: 1 2 3
Linked list: 1 -> 2 -> 3 -> NULL
Reversed linked list: 3 -> 2 -> 1 -> NULL
```

Пункт 2:

```
Enter the number of elements for linked list 1: 4
Enter 4 elements: 1 2 3 9
Enter the number of elements for linked list 2: 4
Enter 4 elements: 1 2 3 10
Linked list 1: 1 -> 2 -> 3 -> 9 -> NULL
Linked list 2: 1 -> 2 -> 3 -> 10 -> NULL
Linked list 1 and Linked list 2 are not identical
```

Пункт 3:

```
Enter num1: 379
Enter num2: 42

Linked list for num1: 9 -> 7 -> 3 -> NULL
Linked list for num2: 2 -> 4 -> NULL
Sum of LL1 and LL2 is: 1 -> 2 -> 4 -> NULL
```

Пункт 4:

```
Binary tree in pre-order traversal:  1 2 5 6 3 8 9 7
Mirrored binary tree in pre-order traversal:  1 3 7 8 9 2 6 5
```

Пункт 5:

```
Binary tree in pre-order traversal:  1 2 5 6 3 8 9 7
Binary tree in pre-order after treeSum:  27 11 5 6 16 9 9 7
```

Завдання № 2 (VNS Lab10. V 22) | Затрачений час – 5-6 год

```
Linked list: Apple <-> Kiwi <-> Banana <-> Pineaplle <-> Orange <-> NULL
Updated linked list after deleting Banana: Apple <-> Kiwi <-> Pineaplle <-> Orange <-> NULL
Updated linked list after adding 2 elements before Apple: Mango <-> Grapes <-> Apple <-> Kiwi <-> Pineaplle <-> Orange <-> NULL
List written to file: list.txt

List deleted successfully.

List restored from file: list.txt
Restored linked list: Mango <-> Grapes <-> Apple <-> Kiwi <-> Pineaplle <-> Orange <-> NULL
PS C:\Users\payar\Epic_6\Code>
```

vns_lab_10_task_1_variant_22_yaryna_panychevska.cpp | list.txt

list.txt

```
1  Mango
2  Grapes
3  Apple
4  Kiwi
5  Pineaplle
6  Orange
7
```

Завдання № 3 (Algotester Lab5. V 3) | Затрачений час – 2-3 год

```
3 4
2 2
1 2 1 0
2 3 2 1
1 2 1 0
```

Compiler	Result	Time (sec.)	Memory (MiB)
C++ 23	Accepted	0.099	7.500

Завдання № 4 (Algotester Lab7-8. V 3) | Затрачений час – 2-3 год

```
11
insert 1
insert 4
insert 5
insert 7
print
1 4 5 7
contains 3
No
size
4
```

Compiler	Result	Time (sec.)	Memory (MiB)
C++ 23	Accepted	0.008	1.422

Висновок:

Впродовж виконання цієї роботи я розібралася як працювати з структурою даних.

Посилання на pull – request: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/625