

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

**Виконав:**

Студент групи ІІІ-12

Бобровицький Олександр Сергійович

**Тема роботи:** Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

**Мета:** ознайомитись з темами лабораторної, опрацювати їх теоретично та навчитися використовувати отриманні знання для вирішення практичних задач

## Теоретичні відомості:

Тема №1 : Основи Динамічних Структур Даних:

- Джерела:
  - <https://acode.com.ua/urok-90-dynamichni-masyvy/>
  - <https://acode.com.ua/urok-89-dynamichne-vydilennya-pam-yati/>
  - <https://acode.com.ua/urok-111-stek-i-kupa/>
- Що опрацьовано:
  - <https://acode.com.ua/urok-90-dynamichni-masyvy/>
  - <https://acode.com.ua/urok-89-dynamichne-vydilennya-pam-yati/>
  - <https://acode.com.ua/urok-111-stek-i-kupa/>
- Статус: Ознайомлений
- Початок опрацювання теми: 15.10
- Звершення опрацювання теми: 15.11

Тема №2 : Зв'язні Списки:

- Джерела:
  - <https://www.geeksforgeeks.org/linked-list-data-structure/>
  - <https://www.geeksforgeeks.org/cpp-linked-list/>
  - <https://www.programiz.com/dsa/linked-list>
- Що опрацьовано:
  - <https://www.geeksforgeeks.org/linked-list-data-structure/>
  - <https://www.geeksforgeeks.org/cpp-linked-list/>
  - <https://www.programiz.com/dsa/linked-list>
- Статус: Ознайомлений
- Початок опрацювання теми: 10.11
- Звершення опрацювання теми: 24.11

Тема №3 : Дерева:

- Джерела:
  - <https://www.geeksforgeeks.org/introduction-to-binary-tree/>
  - <https://www.geeksforgeeks.org/tree-c-cpp-programs/>
- Що опрацьовано:
  - <https://www.geeksforgeeks.org/file-handling-c-classes/>
  - [https://www.w3schools.com/cpp/cpp\\_files.asp](https://www.w3schools.com/cpp/cpp_files.asp)
- Статус: Ознайомлений
- Початок опрацювання теми: 10.11

- Звершення опрацювання теми: 25.11

## Виконання роботи:

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

#### Завдання №1 Algotester Lab 5

- 2 варіант
- Деталі завдання:
  - В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це  $N$ , ширина -  $M$ . Всередині печери є пустота, пісок та каміння. Пустота позначається буквою  $O$ , пісок  $S$  і каміння  $X$ ;
  - Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння. Ваше завдання сказати як буде виглядати печера після землетрусу.

#### Завдання №2 VNS Lab 10

- 11 варіант
- Деталі завдання:
  - Записи в лінійному списку містять ключове поле типу `*char`(рядок символів). Сформувати двонаправлений список. Знищити з нього елемент із заданим ключем, додати елемент із зазначеним номером.

#### Завдання №3 Algotester Lab 7-8 v1

- 1 варіант
- Деталі завдання:
  - Ваше завдання - власноруч реалізувати структуру даних "Двовз'язний список".  
Ви отримаєте  $Q$  запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.
  - Вам будуть поступати запити такого типу: Вставка, Видалення, Визначення розміру, Отримання значення  $i$ -го елемента, Модифікація значення  $i$ -го елемента, Вивід списку на екран.
  - **Для того щоб отримати 50% балів за лабораторну достатньо написати свою структуру.**

#### Завдання №4 Algotester Lab 7-8 v2

- 1 варіант
- Деталі завдання:
  - Ваше завдання - власноруч реалізувати структуру даних "Двовз'язний список".  
Ви отримаєте  $Q$  запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.

- Вам будуть поступати запити такого типу: Вставка, Видалення, Визначення розміру, Отримання значення *i*-го елемента, Модифікація значення *i*-го елемента, Вивід списку на екран.
- Для отримання **100% балів ця структура має бути написана як шаблон класу, у якості параметру використати *int*. Використовувати STL заборонено.**

#### Завдання №5 Class practice work

- 1 варіант
- Деталі завдання:
  - Реалізувати метод реверсу списку.
  - реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі.
  - реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку молодший розряд числа записано в голові списку.
  - реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева
  - реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів

#### Завдання №6 Self practice work

- 1172 варіант
- Деталі завдання:
  - Зеник і Марічка мають масив з *n* цілих чисел *a<sub>i</sub>*. Вони хочуть перевпорядкувати елементи масиву так, щоб сума будь-яких двох послідовних елементів була непарною. Будь ласка, допоможіть їм зробити це або скажіть, що це неможливо.

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

Програма №1 VNS Lab 10

Планований час на реалізацію – 90min

Програма №2 Algotester Lab 5

Планований час на реалізацію – 20min

Програма №3 Algotester Lab 7-8 v1

Планований час на реалізацію – 90min

Програма №4 Algotester Lab 7-8 v2

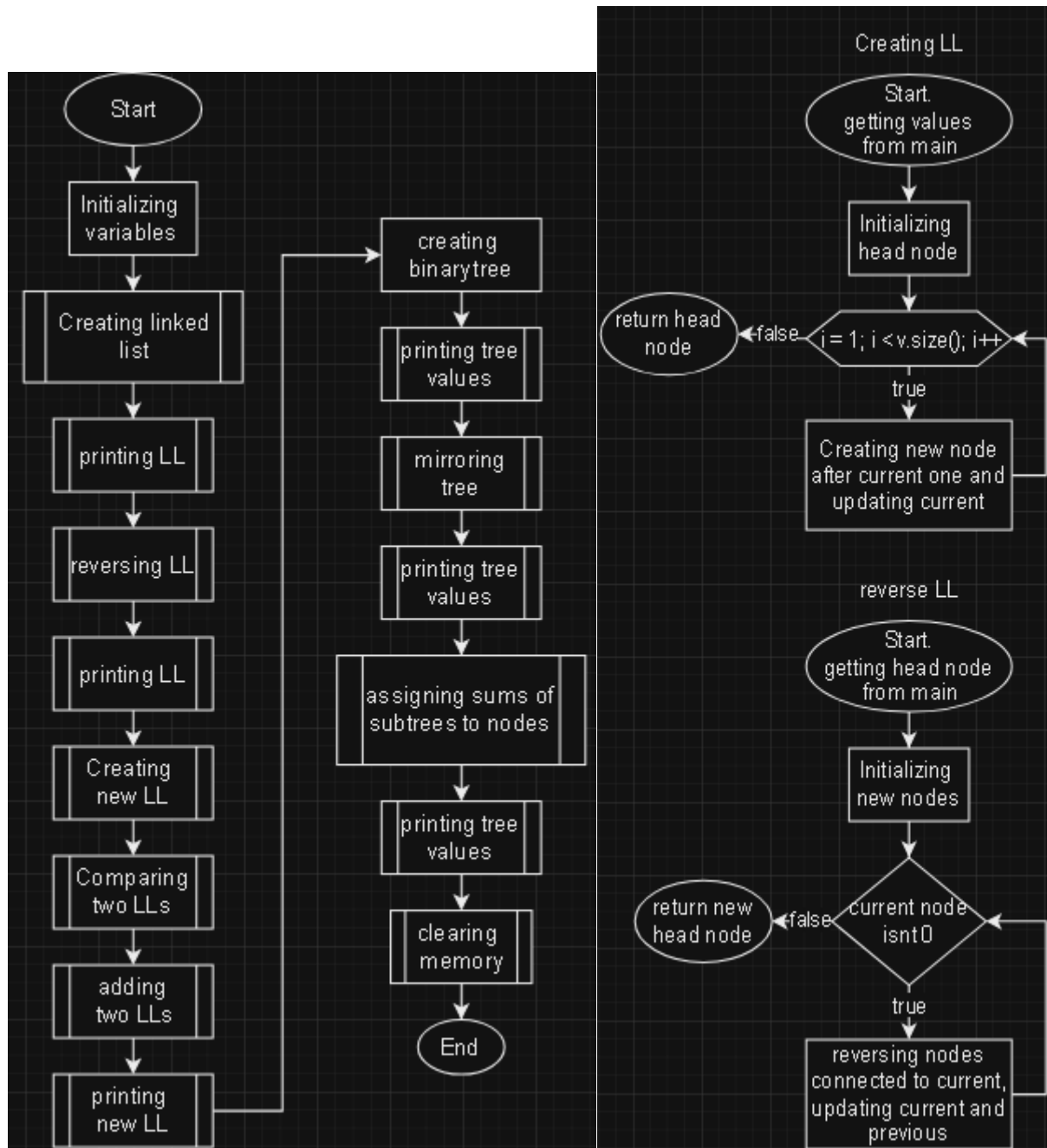
Планований час на реалізацію – 90min

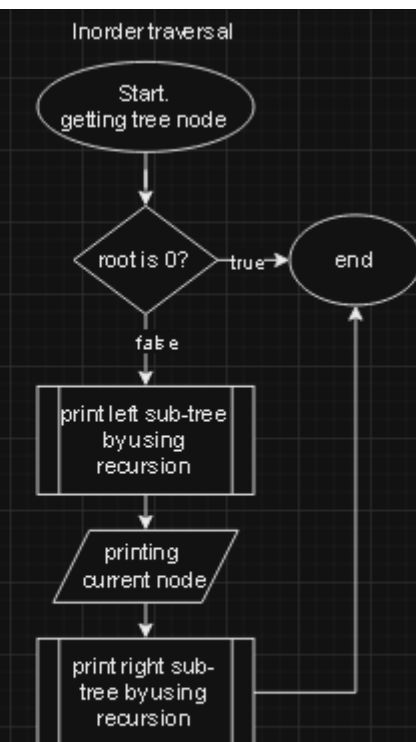
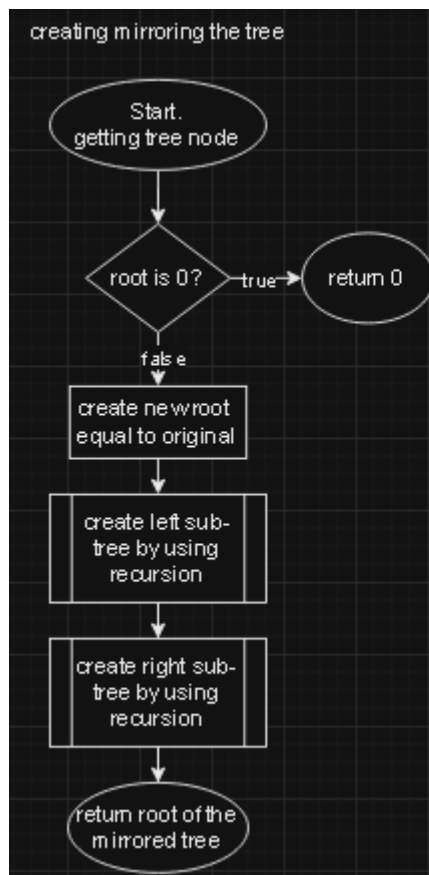
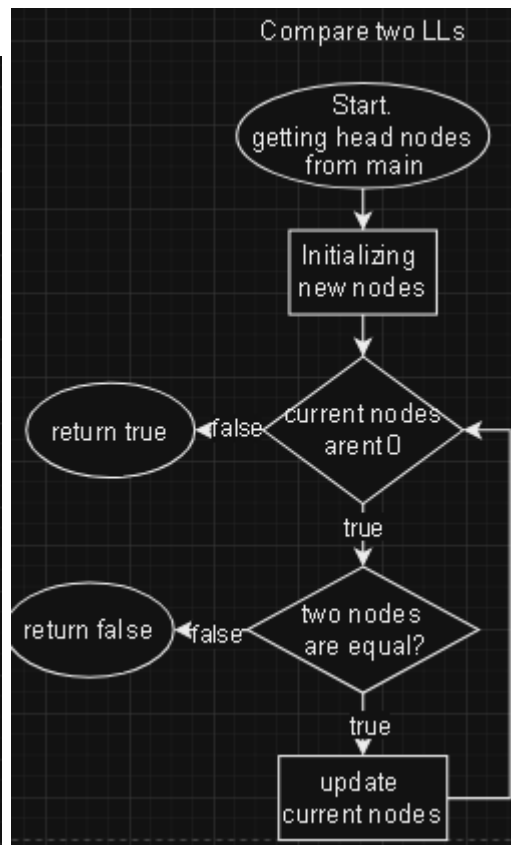
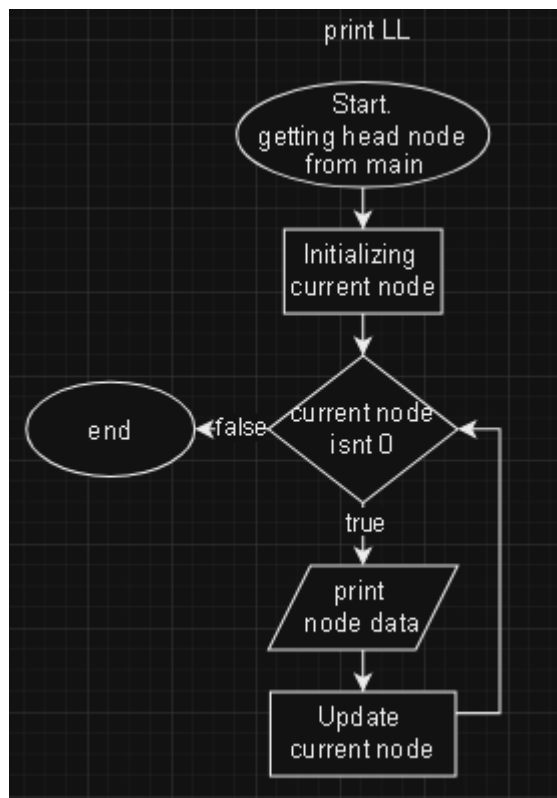
Програма №5 Class practice work

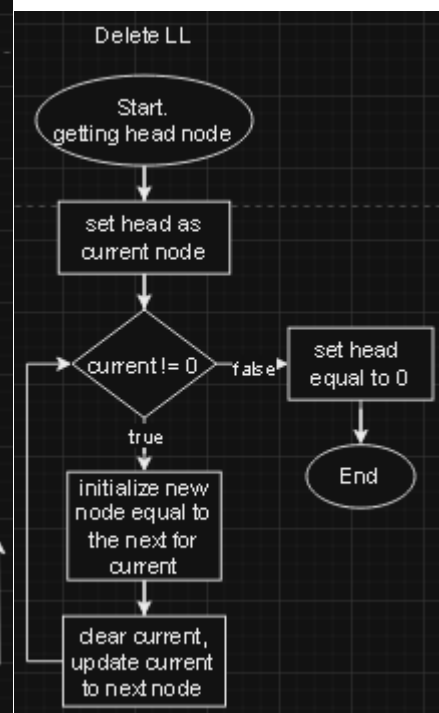
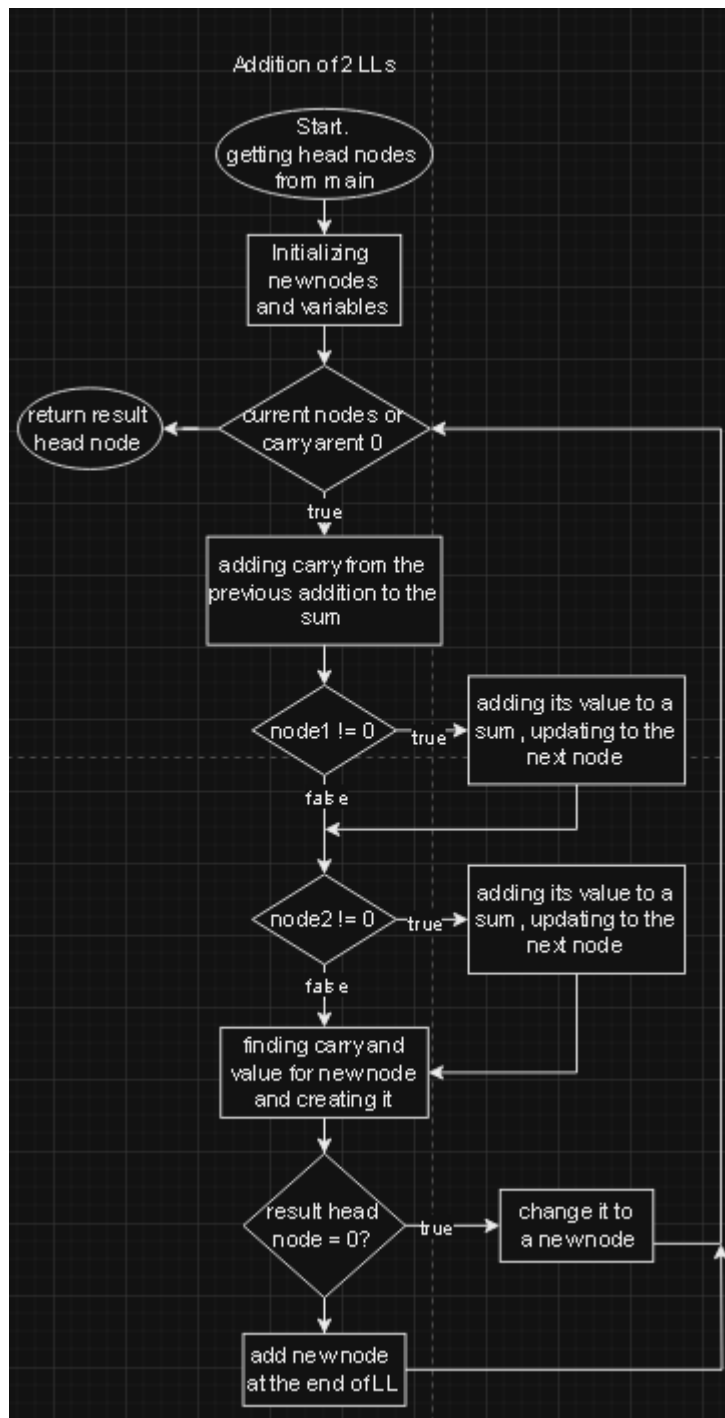
Планований час на реалізацію – 90min

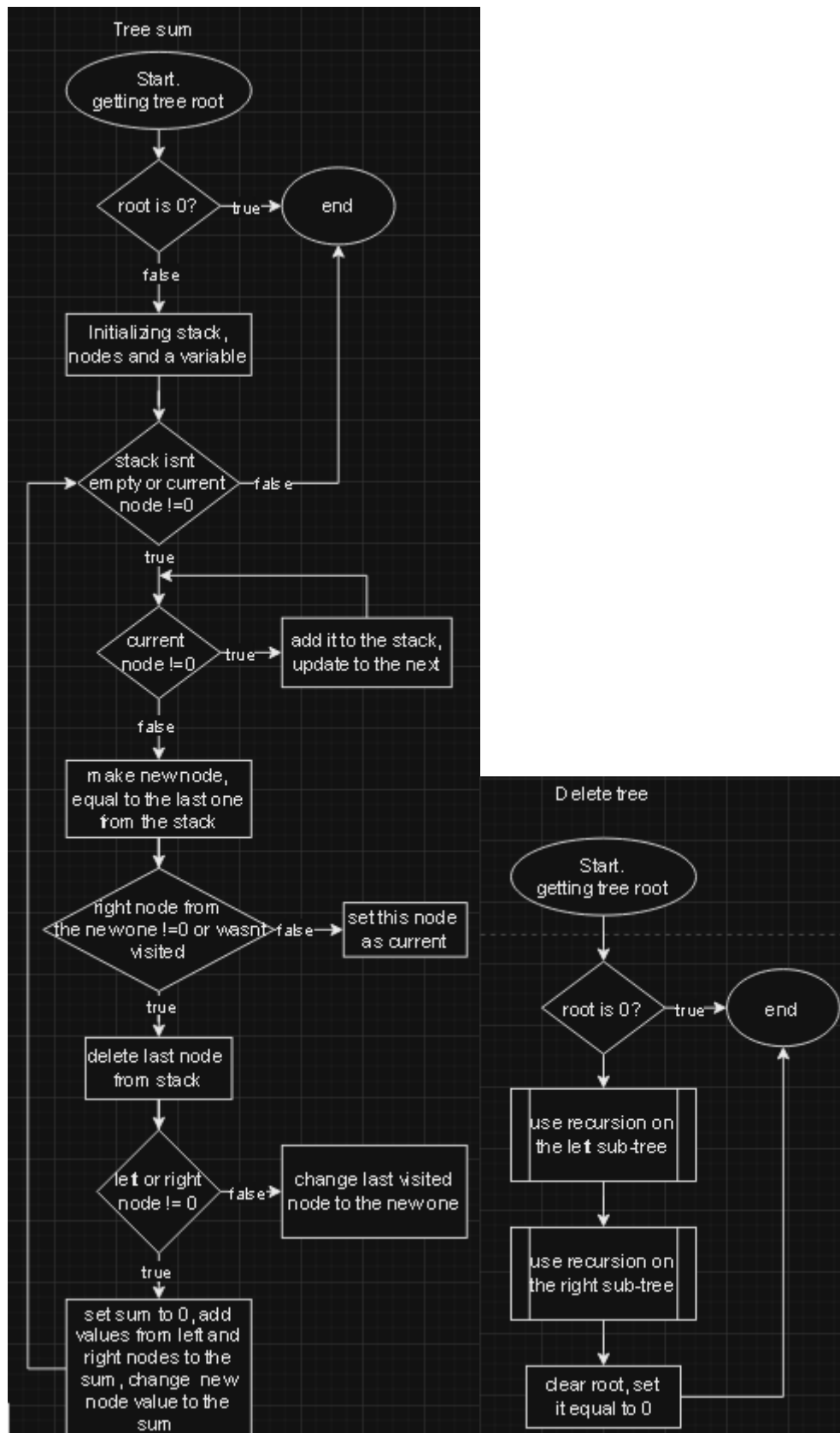
Програма №6 Self -practice work

Блок-схема:









Планований час на реалізацію – 25min



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

Завдання №1

Посилання на файл програми у пул-запиті GitHub [https://github.com/artificial-](https://github.com/artificial-intelligence-)  
[intelligence-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-4e457b528fa7afa68927a632df1f4e012e0b2d1641d8175ddfc8e7b5658057fd)

[department/ai\\_programming\\_playground\\_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-4e457b528fa7afa68927a632df1f4e012e0b2d1641d8175ddfc8e7b5658057fd)

[4e457b528fa7afa68927a632df1f4e012e0b2d1641d8175ddfc8e7b5658057fd](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-4e457b528fa7afa68927a632df1f4e012e0b2d1641d8175ddfc8e7b5658057fd)

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 struct Node
6 {
7     string data;
8     Node *next;
9     Node *prev;
10
11     Node(string d) : data(d), next(nullptr), prev(nullptr) {}
12 };
13
14 Node *createlist()
15 {
16     return nullptr;
17 }
18
19 void addElement(Node *head, string data, int position)
20 {
21     Node *NewNode = new Node(data);
22     if (head == nullptr)
23     {
24         head = NewNode;
25         return;
26     }
27
28     if (position == 0)
29     {
30         NewNode->next = head;
31         head->prev = NewNode;
32         head = NewNode;
33         return;
34     }
35
36     Node *temp = head;
37     int index = 0;
38
39     while (temp->next && index < position - 1)
40     {
41         temp = temp->next;
42         index++;
43     }
44
45     NewNode->next = temp->next;
46     NewNode->prev = temp;
47     if (temp->next)
48     {
49         temp->next->prev = NewNode;
50     }
51     temp->next = NewNode;
52 }
53
54 void printlist(Node *head)
55 {
56     if (head == nullptr)
57     {
58         cout << "list is empty" << endl;
59         return;
60     }
61
62     Node *temp = head;
63     while (temp != nullptr)
64     {
65         cout << temp->data << " ";
66         temp = temp->next;
67     }
68     cout << endl;
69 }
70
71 void deleteElement(Node *head, string key)
72 {
73     if (head == nullptr)
74     {
75         cout << "list is empty" << endl;
76         return;
77     }
78
79     Node *temp = head;
80
81     while (temp->data != key)
82     {
83         temp = temp->next;
84     }
85
86     if (temp == nullptr)
87     {
88         cout << "No element is found for key: " << key << endl;
89         return;
90     }
91
92     if (temp->prev != nullptr)
93         temp->prev->next = temp->next;
94     if (temp->next != nullptr)
95         temp->next->prev = temp->prev;
96     if (temp == head)
97         head = temp->next;
98
99     delete temp;
100
101 }

```

```

1 void writeToFile(Node *head, string filename)
2 {
3     ofstream file;
4     file.open(filename);
5
6     if (!file.is_open())
7     {
8         cout << "Error: unable to open the file" << endl;
9         return;
10    }
11
12    Node *temp = head;
13    while (temp != nullptr)
14    {
15        file << temp->data << endl;
16        temp = temp->next;
17    }
18
19    file.close();
20 }
21
22 void deletelist(Node *head)
23 {
24     if (head == nullptr)
25     {
26         cout << "list is empty" << endl;
27         return;
28     }
29
30     Node *temp = head;
31     while (temp != nullptr)
32     {
33         Node *NextNode = temp->next;
34         delete temp;
35         temp = NextNode;
36     }
37     cout << "list is deleted" << endl;
38     head = nullptr;
39 }
40
41 Node *restoreFromFile(string filename)
42 {
43     string key;
44
45     ifstream file;
46     file.open(filename);
47
48     if (!file.is_open())
49     {
50         cout << "Error: unable to open the file" << endl;
51         return nullptr;
52     }
53
54     Node *head = nullptr;
55     Node *current = nullptr;
56
57     while (getline(file, key))
58     {
59         Node *newNode = new Node(key);
60         if (head == nullptr)
61         {
62             head = current = newNode;
63         }
64         else
65         {
66             current->next = newNode;
67             newNode->prev = current;
68             current = newNode;
69         }
70     }
71
72     cout << "list is restored " << endl;
73
74     file.close();
75
76     return head;
77 }
78
79 int main()
80 {
81     Node *head = createlist();
82     string key, filename;
83     int pos;
84
85     filename = "C:/Users/sabob/projects/ai_programming_playground_2024/ai_12/oleksandr_bobrovyskiy/epic_6/list.txt";
86
87     addElement(head, "a", 0);
88     addElement(head, "b", 1);
89     addElement(head, "c", 2);
90     addElement(head, "d", 3);
91     cout << "Original list: " << endl;
92     printlist(head);
93
94     cout << "Enter a key to delete elements containing it: ";
95     cin >> key;
96
97     deleteElement(head, key);
98
99     cout << "list after deletion of elements containing key: " << endl;
100    printlist(head);
101
102    cout << "Enter the position for inserting new element: ";
103    cin >> pos;
104
105    addElement(head, "e", pos);
106
107    cout << "list after adding new element: " << endl;
108    printlist(head);
109
110    writeToFile(head, filename);
111
112    deletelist(head);
113    printlist(head);
114
115    head = restoreFromFile(filename);
116    printlist(head);
117
118    deletelist(head);
119
120    return 0;
121 }

```

Завдання №2

Посилання на файл програми у пул-запиті GitHub [https://github.com/artificial-intelligence-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-3b57c2bb5741061a6519e22ac5e1379e7fafa1281ec6c3d8e0c0bfa42c17edfd)

[department/ai\\_programming\\_playground\\_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-3b57c2bb5741061a6519e22ac5e1379e7fafa1281ec6c3d8e0c0bfa42c17edfd)

[3b57c2bb5741061a6519e22ac5e1379e7fafa1281ec6c3d8e0c0bfa42c17edfd](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-3b57c2bb5741061a6519e22ac5e1379e7fafa1281ec6c3d8e0c0bfa42c17edfd)

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  int main()
6  {
7      int N, M, emptyRow;
8
9      cin >> N >> M;
10
11     if(N < 1 || N > 1000 || M < 1 || M > 1000)
12     {
13         return 1;
14     }
15
16     vector<string> cave(N);
17
18     for(int i = 0; i < N; i++)
19     {
20         cin >> cave[i];
21         if(cave[i].size() != M)
22         {
23             return 1;
24         }
25     }
26
27     for(int j = 0; j < M; j++)
28     {
29         emptyRow = N - 1;
30         for(int i = N - 1; i >= 0; i--)
31         {
32             if(cave[i][j] == 'X')
33             {
34                 emptyRow = i - 1;
35             }
36             else if(cave[i][j] == 'S')
37             {
38                 cave[i][j] = 'O';
39                 cave[emptyRow][j] = 'S';
40                 emptyRow--;
41             }
42         }
43     }
44
45     for(int i = 0; i < N; i++)
46     {
47         cout << cave[i] << endl;
48     }
49
50     return 0;
51 }
```

Посилання на файл програми у пул-запиті GitHub [https://github.com/artificial-intelligence-department/ai\\_programming\\_playground\\_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-27b62119ca1dc77ffd766b2e655acee3b78b24ebf41bfd297c04871cea4ee172](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-27b62119ca1dc77ffd766b2e655acee3b78b24ebf41bfd297c04871cea4ee172)

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  struct Node
6  {
7      T data;
8      Node *next;
9      Node *prev;
10
11      Node(T d) : data(d), next(nullptr), prev(nullptr) {}
12  };
13
14  class DoublyLinkedList
15  {
16  private:
17      Node *head;
18      Node *tail;
19      int size;
20
21  public:
22      DoublyLinkedList() : head(nullptr), tail(nullptr), size(0) {}
23
24      void insert(vector<int> &data, int position)
25      {
26          Node *current = head;
27          Node *previous = nullptr;
28          int index = 0;
29
30          for (int i = 0; i < position && current; i++)
31          {
32              previous = current;
33              current = current->next;
34          }
35
36          for (int i = 0; i < data.size(); i++)
37          {
38              Node *NewNode = new Node(data[i]);
39
40              if (!previous)
41              {
42                  NewNode->next = head;
43                  if (head)
44                  {
45                      head->prev = NewNode;
46                  }
47                  head = NewNode;
48                  previous = head;
49              }
50              else
51              {
52                  NewNode->next = current;
53                  NewNode->prev = previous;
54                  previous->next = NewNode;
55                  if (current != nullptr)
56                  {
57                      current->prev = NewNode;
58                  }
59
60                  previous = NewNode;
61              }
62              size++;
63          }
64
65          if (current == nullptr)
66          {
67              tail = previous;
68          }
69      }
70
71      void erase(int index, int num)
72      {
73          Node *current = head;
74          for (int i = 0; i < index && current != nullptr; i++)
75          {
76              current = current->next;
77          }
78
79          for (int i = 0; i < num && current != nullptr; i++)
80          {
81              Node *Next = current->next;
82
83              if (current->prev != nullptr)
84              {
85                  current->prev->next = Next;
86              }
87              else
88              {
89                  head = Next;
90              }
91
92              if (Next != nullptr)
93              {
94                  Next->prev = current->prev;
95              }
96              else
97              {
98                  tail = current->prev;
99              }
100
101              delete current;
102              current = Next;
103              size--;
104          }
105      }

```

```

1  int getSize()
2  {
3      return size;
4  }
5
6  int get(int index)
7  {
8      Node *current = head;
9      for (int i = 0; i < index && current != nullptr; i++)
10     {
11         current = current->next;
12     }
13     return current->data;
14 }
15
16 void set(int index, int NewValue)
17 {
18     Node *current = head;
19     for (int i = 0; i < index && current != nullptr; i++)
20     {
21         current = current->next;
22     }
23
24     current->data = NewValue;
25 }
26
27 friend ostream &operator<<(ostream &os, const DoublyLinkedList &list)
28 {
29     Node *current = list.head;
30     while (current != nullptr)
31     {
32         os << current->data << " ";
33         current = current->next;
34     }
35     return os;
36 }
37
38 ~DoublyLinkedList()
39 {
40     Node *current = head;
41     while (current)
42     {
43         Node *nextNode = current->next;
44         delete current;
45         current = nextNode;
46     }
47 }
48 };
49
50 int main()
51 {
52     DoublyLinkedList list;
53     int num, index, value;
54     int numOfActions;
55     string command;
56
57     cin >> numOfActions;
58
59     for (int i = 0; i < numOfActions; i++)
60     {
61         cin >> command;
62
63         if (command == "insert")
64         {
65             cin >> index;
66             cin >> num;
67             vector<int> values(num);
68             for (int i = 0; i < num; i++)
69             {
70                 cin >> values[i];
71             }
72             list.insert(values, index);
73         }
74         else if (command == "erase")
75         {
76             cin >> index >> num;
77             list.erase(index, num);
78         }
79         else if (command == "size")
80         {
81             cout << list.getSize() << endl;
82         }
83         else if (command == "get")
84         {
85             cin >> index;
86             cout << list.get(index) << endl;
87         }
88         else if (command == "set")
89         {
90             cin >> index >> value;
91             list.set(index, value);
92         }
93         else if (command == "print")
94         {
95             cout << list;
96         }
97     }
98
99     return 0;
100 }

```

Завдання №4

Посилання на файл програми у пул-запиті GitHub [https://github.com/artificial-intelligence-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-a925b2c8df7ea7e0a61e03a7a39027716f48b8c4485693df19fe6843d6f50ea8)

[department/ai\\_programming\\_playground\\_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-a925b2c8df7ea7e0a61e03a7a39027716f48b8c4485693df19fe6843d6f50ea8)

[a925b2c8df7ea7e0a61e03a7a39027716f48b8c4485693df19fe6843d6f50ea8](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47ccee460642e5859e96c1#diff-a925b2c8df7ea7e0a61e03a7a39027716f48b8c4485693df19fe6843d6f50ea8)

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 template<typename T>
6 struct Node
7 {
8     T data;
9     Node *next;
10    Node *prev;
11
12    Node(T d) : data(d), next(nullptr), prev(nullptr) {}
13 };
14
15 template<typename T>
16 class DoublyLinkedList
17 {
18 private:
19     Node<T> *head;
20     Node<T> *tail;
21     int size;
22
23 public:
24     DoublyLinkedList() : head(nullptr), tail(nullptr), size(0) {}
25
26     void insert(vector<int> &data, int position)
27     {
28         Node<T> *current = head;
29         Node<T> *previous = nullptr;
30         int index = 0;
31
32         for (int i = 0; i < position && current; i++)
33         {
34             previous = current;
35             current = current->next;
36         }
37
38         for (int i = 0; i < data.size(); i++)
39         {
40             Node<T> *NewNode = new Node(data[i]);
41
42             if (!previous)
43             {
44                 NewNode->next = head;
45                 if (head)
46                 {
47                     head->prev = NewNode;
48                 }
49                 head = NewNode;
50                 previous = head;
51             }
52             else
53             {
54                 NewNode->next = current;
55                 NewNode->prev = previous;
56                 previous->next = NewNode;
57                 if (current != nullptr)
58                 {
59                     current->prev = NewNode;
60                 }
61
62                 previous = NewNode;
63             }
64             size++;
65         }
66
67         if (current == nullptr)
68         {
69             tail = previous;
70         }
71     }
72
73     void erase(int index, int num)
74     {
75         Node<T> *current = head;
76         for (int i = 0; i < index && current != nullptr; i++)
77         {
78             current = current->next;
79         }
80
81         for (int i = 0; i < num && current != nullptr; i++)
82         {
83             Node<T> *Next = current->next;
84
85             if (current->prev != nullptr)
86             {
87                 current->prev->next = Next;
88             }
89             else
90             {
91                 head = Next;
92             }
93
94             if (Next != nullptr)
95             {
96                 Next->prev = current->prev;
97             }
98             else
99             {
100                tail = current->prev;
101            }
102
103            delete current;
104            current = Next;
105            size--;
106        }
107    }
108
109     int getSize()
110     {
111         return size;
112     }

```

```

1
2     int get(int index)
3     {
4         Node<T> *current = head;
5         for (int i = 0; i < index && current != nullptr; i++)
6         {
7             current = current->next;
8         }
9         return current->data;
10    }
11
12    void set(int index, int NewValue)
13    {
14        Node<T> *current = head;
15        for (int i = 0; i < index && current != nullptr; i++)
16        {
17            current = current->next;
18        }
19
20        current->data = NewValue;
21    }
22
23    friend ostream &operator<<(ostream &os, const DoublyLinkedList &list)
24    {
25        Node<T> *current = list.head;
26        while (current != nullptr)
27        {
28            os << current->data << " ";
29            current = current->next;
30        }
31        return os;
32    }
33
34    ~DoublyLinkedList()
35    {
36        Node<T> *current = head;
37        while (current)
38        {
39            Node<T> *nextNode = current->next;
40            delete current;
41            current = nextNode;
42        }
43    }
44 };
45
46 int main()
47 {
48     DoublyLinkedList<int> list;
49     int num, index, value;
50     int numofActions;
51     string command;
52
53     cin >> numofActions;
54
55     for (int i = 0; i < numofActions; i++)
56     {
57         cin >> command;
58
59         if (command == "insert")
60         {
61             cin >> index;
62             cin >> num;
63             vector<int> values(num);
64             for (int i = 0; i < num; i++)
65             {
66                 cin >> values[i];
67             }
68             list.insert(values, index);
69         }
70         else if (command == "erase")
71         {
72             cin >> index >> num;
73             list.erase(index, num);
74         }
75         else if (command == "size")
76         {
77             cout << list.getSize() << endl;
78         }
79         else if (command == "get")
80         {
81             cin >> index;
82             cout << list.get(index) << endl;
83         }
84         else if (command == "set")
85         {
86             cin >> index >> value;
87             list.set(index, value);
88         }
89         else if (command == "print")
90         {
91             cout << list;
92         }
93     }
94
95     return 0;
96 }

```



Завдання №5

Посилання на файл програми у пул-запиті GitHub [https://github.com/artificial-](https://github.com/artificial-intelligence-)  
[intelligence-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-62d98e65c7e565c169990cd295e5d45e87d8bd0902d2eaf7352dc44e9ee49613)

[department/ai\\_programming\\_playground\\_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-62d98e65c7e565c169990cd295e5d45e87d8bd0902d2eaf7352dc44e9ee49613)

[62d98e65c7e565c169990cd295e5d45e87d8bd0902d2eaf7352dc44e9ee49613](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-62d98e65c7e565c169990cd295e5d45e87d8bd0902d2eaf7352dc44e9ee49613)

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 struct Node
6 {
7     int data;
8     Node *next;
9
10    Node(int value) : data(value), next(nullptr) {}
11 };
12
13 struct TreeNode
14 {
15     int data;
16     TreeNode *left;
17     TreeNode *right;
18
19    TreeNode(int value) : data(value), left(nullptr), right(nullptr) {}
20 };
21
22 void deleteList(Node *head)
23 {
24     Node *current = head;
25     while (current != nullptr)
26     {
27         Node *next = current->next;
28         delete current;
29         current = next;
30     }
31     head = nullptr;
32 }
33
34 void deleteTree(TreeNode *root)
35 {
36     if (root == nullptr)
37     {
38         return;
39     }
40
41     deleteTree(root->left);
42     deleteTree(root->right);
43
44     delete root;
45     root = nullptr;
46 }
47
48 Node *createList(vector<int> v)
49 {
50     Node *head = new Node(v[0]);
51     Node *current = head;
52
53     for (int i = 1; i < v.size(); i++)
54     {
55         current->next = new Node(v[i]);
56         current = current->next;
57     }
58
59     return head;
60 }
61
62 Node *reverse(Node *head)
63 {
64     Node *previous = nullptr;
65     Node *next = nullptr;
66     Node *current = head;
67
68     while (current != nullptr)
69     {
70         next = current->next;
71         current->next = previous;
72         previous = current;
73         current = next;
74     }
75
76     return previous;
77 }
78
79 void printList(Node *head)
80 {
81     Node *temp = head;
82     while (temp != nullptr)
83     {
84         cout << temp->data << " ";
85         temp = temp->next;
86     }
87 }
88
89 bool compare(Node *h1, Node *h2)
90 {
91     Node *temp1 = h1;
92     Node *temp2 = h2;
93     while (temp1 != nullptr && temp2 != nullptr)
94     {
95         if ((temp1->data != temp2->data) || (temp1 == nullptr && temp2 != nullptr) || (temp2 == nullptr && temp1 != nullptr))
96         {
97             return false;
98         }
99
100         temp1 = temp1->next;
101         temp2 = temp2->next;
102     }
103
104     return true;
105 }
106
107 Node *add(Node *n1, Node *n2)
108 {
109     Node *resultHead = nullptr;
110     Node *resultTail = nullptr;
111     Node *temp1 = n1;
112     Node *temp2 = n2;
113     int num = 0, carry = 0, sum = 0;
114
115     while (temp1 != nullptr || temp2 != nullptr || carry != 0)
116     {
117         sum = carry;
118         if (temp1 != nullptr)
119         {
120             sum += temp1->data;
121             temp1 = temp1->next;
122         }
123
124         if (temp2 != nullptr)
125         {
126             sum += temp2->data;
127             temp2 = temp2->next;
128         }
129
130         carry = sum / 10;
131         num = sum % 10;
132
133         Node *newNode = new Node(num);
134
135         if (resultHead == nullptr)
136         {
137             resultHead = newNode;
138         }
139         else
140         {
141             resultTail->next = newNode;
142         }
143         resultTail = newNode;
144     }
145
146     return resultHead;
147 }
148

```

```

1
2 TreeNode *create_mirror_flip(TreeNode *root)
3 {
4     if (root == nullptr)
5     {
6         return nullptr;
7     }
8
9     TreeNode *mirrored = new TreeNode(root->data);
10
11     mirrored->left = create_mirror_flip(root->right);
12     mirrored->right = create_mirror_flip(root->left);
13
14     return mirrored;
15 }
16
17 void inorderTraversal(TreeNode *root)
18 {
19     if (root == nullptr)
20         return;
21
22     inorderTraversal(root->left);
23     cout << root->data << " ";
24     inorderTraversal(root->right);
25 }
26
27 void tree_sum(TreeNode *root)
28 {
29     if (root == nullptr)
30         return;
31
32     stack<TreeNode*> NodeStack;
33     TreeNode *lastVisited = nullptr;
34     TreeNode *current = root;
35     int sum;
36
37     while (!NodeStack.empty() || current != nullptr)
38     {
39         while (current != nullptr)
40         {
41             NodeStack.push(current);
42             current = current->left;
43         }
44
45         TreeNode *topNode = NodeStack.top();
46
47         if (topNode->right == nullptr || lastVisited == topNode->right)
48         {
49             NodeStack.pop();
50
51             if (topNode->left != nullptr || topNode->right != nullptr)
52             {
53                 sum = 0;
54                 sum += (topNode->left != nullptr) ? topNode->left->data : 0;
55                 sum += (topNode->right != nullptr) ? topNode->right->data : 0;
56                 topNode->data = sum;
57             }
58
59             lastVisited = topNode;
60         }
61         else
62         {
63             current = topNode->right;
64         }
65     }
66 }
67
68 int main()
69 {
70     int n = 6;
71     vector<int> values1 = {4, 6, 9};
72     vector<int> values2 = {3, 2, 5};
73
74     Node *head = createlist(values1);
75
76     cout << "Original list: ";
77     printlist(head);
78
79     head = reverse(head);
80
81     cout << "\nReversed list: ";
82     printlist(head);
83
84     Node *head2 = createlist(values2);
85
86     cout << "\nResult of the comparison: " << compare(head, head2) << endl;
87
88     Node *additionHead = add(head, head2);
89
90     printlist(additionHead);
91
92     TreeNode *root = new TreeNode(1);
93     root->left = new TreeNode(2);
94     root->right = new TreeNode(3);
95     root->left->left = new TreeNode(4);
96     root->left->right = new TreeNode(5);
97     root->right->left = new TreeNode(6);
98     root->right->right = new TreeNode(7);
99
100     cout << "\nInorder traversal of the original tree: " << endl;
101     inorderTraversal(root);
102
103     TreeNode *mirrored = create_mirror_flip(root);
104
105     cout << "\nInorder traversal after mirroring: " << endl;
106     inorderTraversal(mirrored);
107
108     tree_sum(root);
109
110     cout << "\nInorder traversal after finding sum of nodes: " << endl;
111     inorderTraversal(root);
112
113     deletelist(head);
114     deletelist(head2);
115     deletelist(additionHead);
116     deleteTree(mirrored);
117     deleteTree(root);
118
119     return 0;
120 }

```

Завдання №6

Посилання на файл програми у пул-запиті GitHub [https://github.com/artificial-](https://github.com/artificial-intelligence-)  
[intelligence-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-6a55165f4a113b293060c3a9ec97f256da870412844800744ba7440d920acafb)

[department/ai\\_programming\\_playground\\_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-6a55165f4a113b293060c3a9ec97f256da870412844800744ba7440d920acafb)

[6a55165f4a113b293060c3a9ec97f256da870412844800744ba7440d920acafb](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/491/commits/d2422006e96ab147ac47cce460642e5859e96c1#diff-6a55165f4a113b293060c3a9ec97f256da870412844800744ba7440d920acafb)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      cin >> n;
8
9      vector<int> a(n), evens, odds;
10     for (int i = 0; i < n; i++)
11     {
12         cin >> a[i];
13         if (a[i] % 2 == 0)
14         {
15             evens.push_back(a[i]);
16         }
17         else
18         {
19             odds.push_back(a[i]);
20         }
21     }
22
23     if (abs((int)evens.size() - (int)odds.size()) > 1)
24     {
25         cout << -1 << endl;
26         return 0;
27     }
28
29     vector<int> result;
30     bool evenTurn = evens.size() >= odds.size();
31     while (!evens.empty() || !odds.empty())
32     {
33         if (evenTurn && !evens.empty())
34         {
35             result.push_back(evens.back());
36             evens.pop_back();
37         }
38         else if (!evenTurn && !odds.empty())
39         {
40             result.push_back(odds.back());
41             odds.pop_back();
42         }
43         evenTurn = !evenTurn;
44     }
45
46     for(int i = 0; i < result.size(); i++)
47     {
48         cout << result[i] << " ";
49     }
50
51     return 0;
52 }

```

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

Завдання №1 Деталі по виконанню і тестуванню програми

```
Original list:
a b c d
Enter a key to delete elements containing it: a
List after deletion of elements containing key:
b c d
Enter the position for inserting new element: 1
List after adding new element:
b e c d
List is deleted
list is empty
list is restored
b e c d
List is deleted
```

Блок №1 Результат виконання завдання

Час затрачений на виконання завдання – 90min

Завдання №2 Деталі по виконанню і тестуванню програми

```
5 5
SSOSS
OOOOO
SOOXX
OOOOS
OOSOO
OOOOO
OOOSS
OOOXX
SOOOO
SSSOS
```

Блок №1 Результат виконання завдання

Час затрачений на виконання завдання – 25min

Завдання №3 Деталі по виконанню і тестуванню програми

```
9
insert
0
5
1 2 3 4 5

insert
2
3
7 7 7

print
1 2 7 7 7 3 4 5
```

```
erase
1 2

print
1 7 7 3 4 5
size
6

get
3
3

set
3 13

print
1 7 7 13 4 5
```

Блок №1 Результат виконання завдання

Час затрачений на виконання завдання – 90min

Завдання №4 Деталі по виконанню і тестуванню програми

```
9
insert
0
5
1 2 3 4 5

insert
2
3
7 7 7

print
1 2 7 7 7 3 4 5

erase
1 2

print
1 7 7 3 4 5
size
6

get
3
3

set
3 13

print
1 7 7 13 4 5
```

Блок №1 Результат виконання завдання

Час затрачений на виконання завдання – 5min

Завдання №5 Деталі по виконанню і тестуванню програми

```
Original list: 4 6 9
Reversed list: 9 6 4
Result of the comparison: 0
2 9 9
Inorder traversal of the original tree:
4 2 5 1 6 3 7
Inorder traversal after mirroring:
7 3 6 1 5 2 4
Inorder traversal after finding sum of nodes:
4 9 5 22 6 13 7
```

Блок №1 Результат виконання завдання

Час затрачений на виконання завдання – 90min

Завдання №6 Деталі по виконанню і тестуванню програми

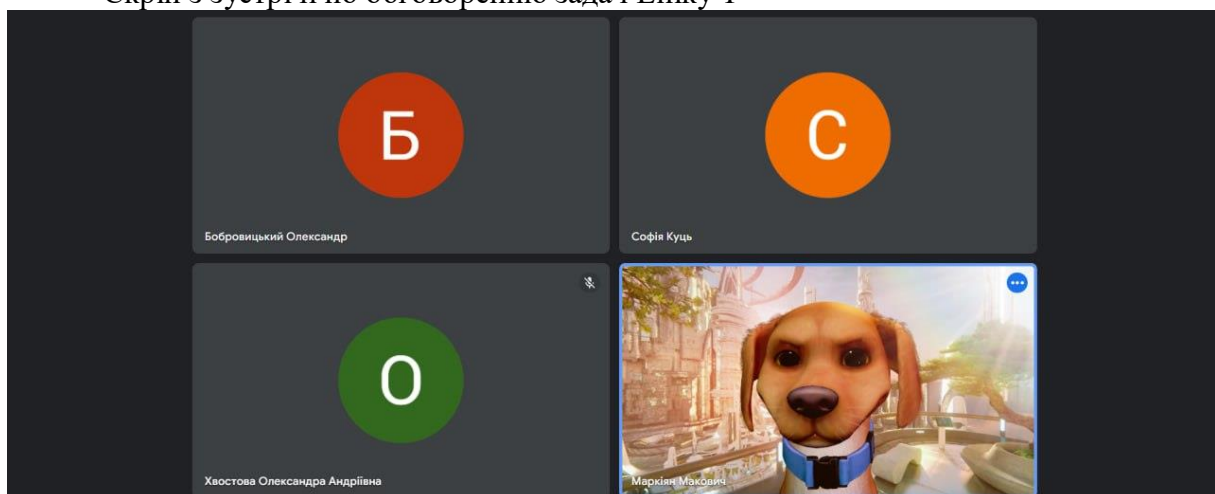
```
3
4 7 47
47 4 7
```

Блок №1 Результат виконання завдання

Час затрачений на виконання завдання – 20min

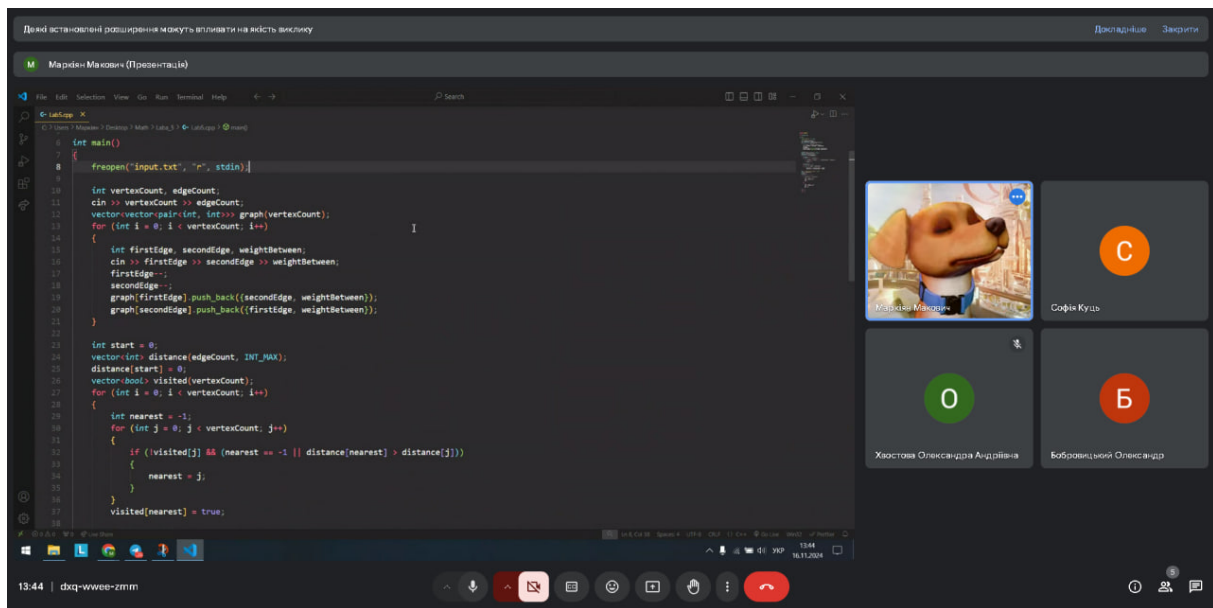
## 5. Кооперація з командою:

- Скрін з зустрічі по обговоренню задач Епіку 1

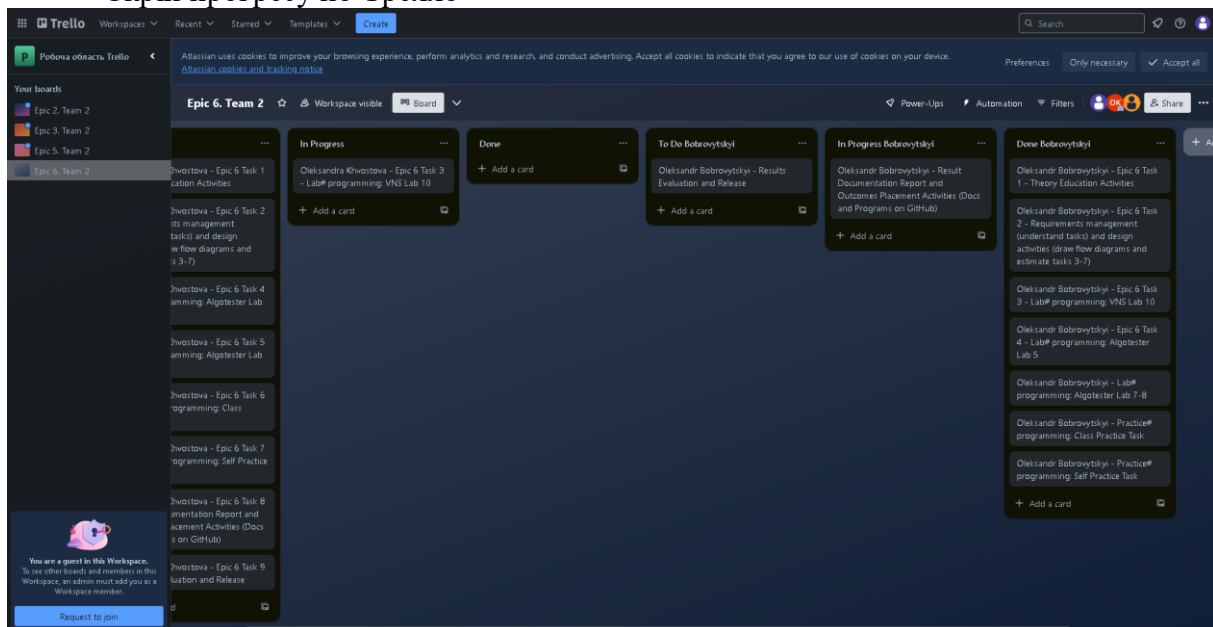


- Скрін з зустрічі по обговоренню задач Епіку 2





## – Скрін прогресу по Трелло



**Висновок:** я ознайомився з темами лабораторної, опрацювати їх теоретично та навчитися використовувати отриманні знання для вирішення практичних задач.