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 ШІ-12
Кутельмах Євген Петрович

Львів 2024

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

Мета: Отримати практичні навички у роботі з динамічними структурами, алгоритмами обробок динамічних структур

Теоретичні відомості:

1) *Стек і купа, Черги*
   https://acode.com.ua/urok-111-stek-i-kupa/
   https://www.youtube.com/watch?v=Yhw8NbjrSFA&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD
   Завдяки цьому уроці, я зрозумів чим відрізняється стек і купа, коли варто використати певний тип даних. Ще на цьому уроці та через іншу інформацію в інтернеті, зокрема дане відео, я дізнався як можна реалізувати стек власноруч, як чергу, та інші види черг. Витратив на це близько години.
2) *Інші динамічні структури*
   https://www.youtube.com/watch?v=-25REjF_atI&t=1189s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD
   На цьому відео я розглянув, як можна створити однозв'язний список, ще виконуючи цю роботу, я розглянув в інтернеті як реалізовуються інші динамічні структури: двозв'язний список, бінарне дерево, динамічний масив, тощо. На це я витратив часу близько 3 годин.

Виконання роботи:

Task 3 - Lab# programming: VNS Lab 10

Завдання: 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.

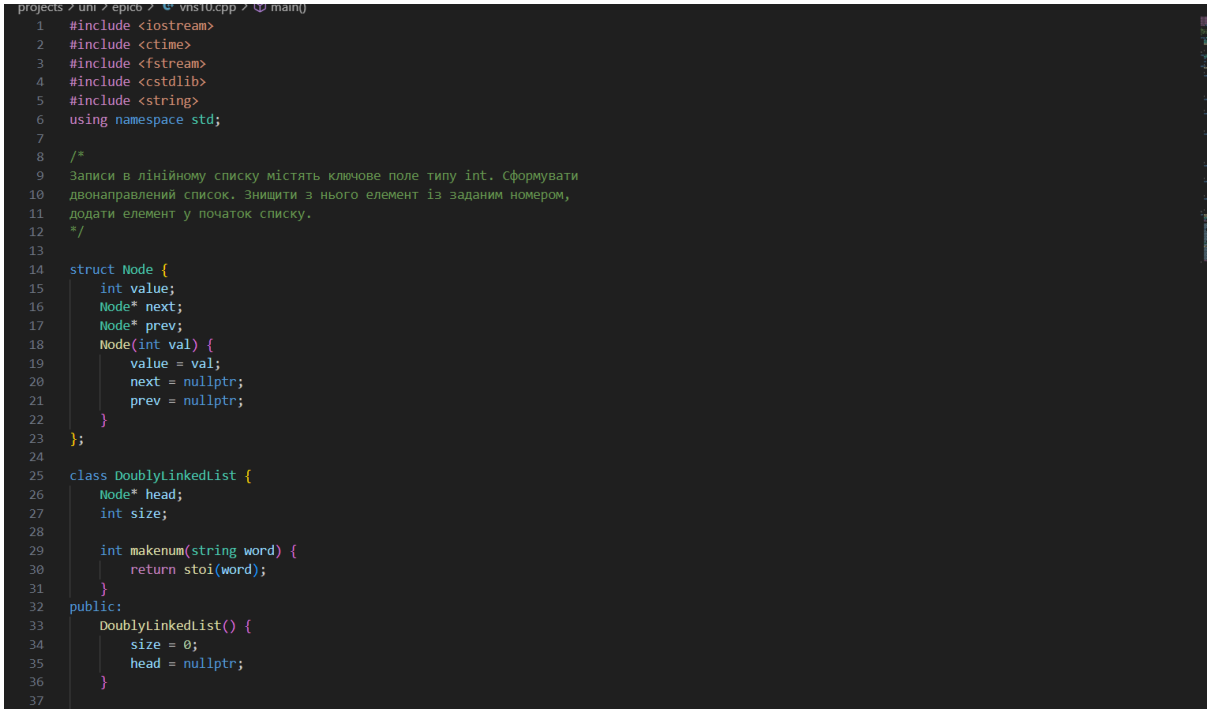
5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

Варіант 6: Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Знищити з нього елемент із заданим номером, додати елемент у початок списку.

Програма:



```cpp
#include <iostream>
#include <ctime>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;

/*
Записи в лінійному списку містять ключове поле типу int. Сформувати
двонаправлений список. Знищити з нього елемент із заданим номером,
додати елемент у початок списку.
*/

struct Node {
    int value;
    Node* next;
    Node* prev;
    Node(int val) {
        value = val;
        next = nullptr;
        prev = nullptr;
    }
};

class DoublyLinkedList {
    Node* head;
    int size;

    int makenum(string word) {
        return stoi(word);
    }
public:
    DoublyLinkedList() {
        size = 0;
        head = nullptr;
    }
```

```cpp
    void pushBack(int value) {
        Node* newNode = new Node(value);
        if(head == nullptr)
            head = newNode;
        else {
            Node* curr = head;
            while(curr->next != nullptr) {
                curr = curr->next;
            }
            curr->next = newNode;
            newNode->prev = curr;
        }
        size++;
    }

    void pushBack(int value, int n) {
        Node* newNode = new Node(value);
        if(size==0) head = newNode;
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
        size++;
    }

    void print() {
        if(head == nullptr)
            cout << "The list is empty";
        else {
            Node* curr = head;
            while(curr != nullptr) {
                cout << curr->value << " ";
                curr = curr->next;
            }
        }
```

```
    void remove(int index) {
        Node *current = head;
        int count = 0;
        while(count != index) {
            current = current->next;
            count++;
        }

        if (current->prev != nullptr)
            current->prev->next = current->next;
        else
            head = current->next;

        if (current->next != nullptr)
            current->next->prev = current->prev;

        delete current;
        size--;
    }

    void write(string name) {
        ofstream file(name);
        Node *cur = head;
        for(int i = 0; i < size; i++) {
            file << cur->value << "\n";
            cur = cur->next;
        }
        file.close();
    }

    void del() {
        Node *cur = head;
        while(cur != nullptr) {
            Node *nextN = cur->next;
            delete cur;
            cur = nextN;
        }
        head = nullptr;
        size = 0;
    }

    void recreate(string name) {
        ifstream file(name);
        string buffer;
        while(getline(file, buffer)) {
            int n = makenum(buffer);
            pushBack(n);
        }
    }
};

int main() {
    srand(static_cast<unsigned int>(time(0)));
    DoublyLinkedList list;
    int len = rand()%6 + 1;
    for(int i = 0; i < len; i++)
        list.pushBack(rand()%15);
    cout << "We have created our list, it contains: \n";
    list.print();
    cout << "Which element you want to delete?(enumaration starts from 0): ";
    int index;
    cin >> index;
    list.remove(index);
```

```cpp
int main() {
    srand(static_cast<unsigned int>(time(0)));
    DoublyLinkedList list;
    int len = rand()%6 + 1;
    for(int i = 0; i < len; i++)
        list.pushBack(rand()%15);
    cout << "We have created our list, it contains: \n";
    list.print();
    cout << "Which element you want to delete?(enumaration starts from 0): ";
    int index;
    cin >> index;
    list.remove(index);
    cout << "Our list after deleting #" << index << " element:\n";
    list.print();
    list.pushBack(rand()%15, 0);
    cout << "Our list after adding a new element at the front:\n";
    list.print();
    cin.ignore();
    string name;
    cout << "Enter a name for the file: ";
    getline(cin, name);
    list.write(name);
    cout << "After writing to file, we delete our list:\n";
    list.del();
    list.print();
    cout << "Now we recreating our list from your file:\n";
    list.recreate(name);
    list.print();
    list.del();
    return 0;
}
```

Результат:

```
We have created our list, it contains:
4 2 14 1
Which element you want to delete?(enumaration starts from 0): 2
Our list after deleting #2 element:
4 2 1
Our list after adding a new element at the front:
1 4 2 1
Enter a name for the file: test
After writing to file, we delete our list:
The list is empty
Now we recreating our list from your file:
1 4 2 1
PS C:\Users\kutel\.vscode\projects\uni\epic6>
```

test:

```
projects > uni > epic6 >  test
1   1
2   4
3   2
4   1
5
```

На це завдання я витратив близько 2 годин часу.

Task 4 - Lab# programming: Algotester Lab 5

Умова:

Lab 5v2

Limits: 1 sec., 256 MiB

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.

Input

У першому рядку 2 цілих числа N та M - висота та ширина печери

У N наступних рядках стрічка row_i яка складається з N цифер - і-й рядок матриці, яка відображає стан печери до землетрусу.

Output

N рядків, які складаються з стрічки розміром M - стан печери після землетрусу.

Constraints

$1 \leq N, M \leq 1000$

$|row_i| = M$

$row_i \in \{X, S, O\}$

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
9 days ago	C++ 23	Accepted	0.027	1.836	View

Showing 1 to 1 of 1 rows

Програма:

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

bool check(char *col, int N) {
    for(int i = 0; i < N - 1; i++) {
        if(col[i] == 'S' && col[i+1] == 'O') return false;
    }
    return true;
}

void dochanges(char *col,int N) {
    auto p = find(col, col + N, 'X');
    if(p == col + N) {
        int s = count(col, col + N, 'S');
        for(int i = N-1; i >= 0; i--) {
            if(s>0) {
                col[i] = 'S';
                s--;
            }
            else col[i] = 'O';
        }
    } else {
        while(!check(col, N)) {
            for(int i = 0; i < N - 1; i++) {
                if(col[i] == 'S' && col[i + 1] == 'O')
                    swap(col[i], col[i + 1]);
            }
        }
    }
}
```

```cpp
void makeCols(char **arr,int N, int M) {
    for(int i = 0; i < M; i++) {
        char *col = new char[N];
        for(int j = 0; j < N; j++) {
            col[j] = arr[j][i];
        }
        dochanges(col, N);
        for(int j = 0; j < N; j++) {
            arr[j][i] = col[j];
        }
        delete[] col;
        col = nullptr;
    }
}

void print(char **arr, int size) {
    for(int i = 0; i < size; i++)
        cout << arr[i] << "\n";
}
```

```
53  int main() {
54      int N, M;
55      cin >> N >> M;
56      char **arr = new char*[N];
57      cin.ignore(23421, '\n');
58      for(int i = 0; i < N; i++) {
59          arr[i] = new char[M+1];
60          cin.getline(arr[i], M + 1);
61      }
62      makeCols(arr, N, M);
63      print(arr, N);
64      for(int i = 0; i < N; i++) {
65          delete[] arr[i];
66          arr[i] = nullptr;
67      }
68      delete[] arr;
69      arr = nullptr;
70      return 0;
71  }
```

Результат:

```
5 5
SSOSS
OOOOO
SOOXX
OOOOS
OOSOO
OOOOO
OOOSS
OOOXX
SOOOO
SSSOS
PS C:\Users\kutel\.vscode\projects\uni\epic6>
```

На дану задачу я витратив 30 хв.

Task 5 - Lab# programming: Algotester Lab 7-8

Умова:

Lab 78v2

Limits: 1 sec., 256 MiB

Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.

Вам будуть поступати запити такого типу:

- **Вставка**:
  Ідентифікатор - $insert$
  Ви отримуєте ціле число $index$ елемента, на місце якого робити вставку.
  Після цього в наступному рядку рядку написане число N - розмір масиву, який треба вставити.
  У третьому рядку N цілих чисел - масив, який треба вставити на позицію $index$.
- **Видалення**:
  Ідентифікатор - $erase$
  Ви отримуєте 2 цілих числа - $index$, індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити.
- **Визначення розміру**:
  Ідентифікатор - $size$
  Ви не отримуєте аргументів.
  Ви виводите кількість елементів у динамічному масиві.
- **Визначення кількості зарезервованої пам'яті**:
  Ідентифікатор - $capacity$
  Ви не отримуєте аргументів.

- **Визначення кількості зарезервованої пам'яті**:
  Ідентифікатор - $capacity$
  Ви не отримуєте аргументів.

  Ви виводите кількість зарезервованої пам'яті у динамічному масиві.
  Ваша реалізація динамічного масиву має мати фактор росту (Growth factor) рівний 2.
- **Отримання значення** i-го елементу
  Ідентифікатор - get
  Ви отримуєте ціле число - $index$, індекс елемента.
  Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора []
- **Модифікація значення** i-го елементу
  Ідентифікатор - set
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора []
- **Вивід динамічного масиву на екран**
  Ідентифікатор - $print$
  Ви не отримуєте аргументів.
  Ви виводите усі елементи динамічного масиву через пробіл.
  Реалізувати використовуючи перегрузку оператора <<

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.

Notes

Гарантується, що усі дані коректні. Виходу за межі масиву або розмір, більший ніж розмір масиву недопустимі.
Індекси починаються з нуля.
Для того щоб отримати 50% балів за лабораторну достатньо написати свою структуру.
Для отримання 100% балів ця структура має бути написана як шаблон класу, у якості параметру використати int.
Використовувати STL заборонено.

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
27 minutes ago	C++ 23	Accepted	0.006	1.434	View
an hour ago	C++ 23	Accepted	0.006	1.340	View
an hour ago	C++ 23	Wrong Answer 1	0.002	1.039	View

Програма(з використанням структури):

```cpp
#include <iostream>
#include <string>
using namespace std;

struct DynamicMas {
private:
    int s;
    int capacity;
    int *array;

    void resize(int size) {
        if(size >= capacity) {
            while(size >= capacity)
                capacity *= 2;
            int *newArray = new int[capacity];
            for (int i = 0; i < s; i++) {
                newArray[i] = array[i];
            }
            delete[] array;
            array = newArray;
        }
    }

public:
    DynamicMas() {
        s = 0;
        capacity = 1;
        array = new int[capacity];
    }

    ~DynamicMas() {
        delete[] array;
    }

    int cap() {
        return capacity;
    }

    void insert(int index, int num, int *array2) {
        resize(s+num);

        for (int i = s - 1; i >= index; i--) {
            array[i + num] = array[i];
        }
        for(int i = 0; i < num; i++) {
            array[index + i] = array2[i];
        }

        s += num;
    }

    void erase(int index, int n) {
        for (int i = index; i < s - n; i++) {
            array[i] = array[i + n];
        }
        s -= n;
    }

    int size() {
        return s;
    }

    int& operator[] (const int index) {
        return array[index];
    }
```

```
66
67      int get(int index) {
68          return array[index];
69      }
70
71      void set(int index, int num) {
72          array[index] = num;
73      }
74
75      friend ostream& operator<< (ostream &out, const DynamicMas &arr) {
76          for(int i = 0; i < arr.s; i++) {
77              out << arr.array[i] << " ";
78          }
79          return out;
80      }
81
82      void print() {
83          cout << *this << "\n";
84      }
85  };
86
87  int main() {
88      int Q;
89      cin >> Q;
90      DynamicMas arr;
91      for(int i = 0; i < Q; i++) {
92          string com;
93          cin >> com;
94
```

```
        if(com == "size") {
            cout << arr.size() << "\n";
        } else if(com == "capacity") {
            cout << arr.cap() << "\n";
        } else if(com == "print") {
            arr.print();
        } else if(com == "get") {
            int num;
            cin >> num;
            cout << arr.get(num) << "\n";
        } else if(com == "set") {
            int a,b;
            cin >> a >> b;
            arr.set(a, b);
        } else if(com == "erase") {
            int a, b;
            cin >> a >> b;
            arr.erase(a, b);
        } else {
            int a, b;
            cin >> a >> b;
            int *ar = new int[b];
            for(int i = 0; i < b; i++)
                cin >> ar[i];
            arr.insert(a, b, ar);
            delete[] ar;
            ar = nullptr;
        }
    }
    return 0;
}
```

Програма(з використанням шаблону класу):

```cpp
#include <iostream>
#include <string>
using namespace std;

template <class T>
class DynamicMas {
private:
    int s;
    int capacity;
    T *array;

    void resize(int size) {
        if(size >= capacity) {
            while(size >= capacity)
                capacity *= 2;
            T *newArray = new T[capacity];
            for (int i = 0; i < s; i++) {
                newArray[i] = array[i];
            }
            delete[] array;
            array = newArray;
        }
    }

public:
    DynamicMas() {
        s = 0;
        capacity = 1;
        array = new T[capacity];
    }

    ~DynamicMas() {
        delete[] array;
    }

    int cap() {
        return capacity;
    }

    void insert(int index, int num, T *array2) {
        resize(s+num);

        for (int i = s - 1; i >= index; i--) {
            array[i + num] = array[i];
        }
        for(int i = 0; i < num; i++) {
            array[index + i] = array2[i];
        }

        s += num;
    }

    void erase(int index, int n) {
        for (int i = index; i < s - n; i++) {
            array[i] = array[i + n];
        }
        s -= n;
    }

    int size() {
        return s;
    }

    T& operator[] (const int index) {
        return array[index];
    }
```

```
    T get(int index) {
        return array[index];
    }

    void set(int index, T num) {
        array[index] = num;
    }

    friend ostream& operator<< (ostream &out, const DynamicMas<T> &arr) {
        for(int i = 0; i < arr.s; i++) {
            out << arr.array[i] << " ";
        }
        return out;
    }

    void print() {
        cout << *this << "\n";
    }
};

int main() {
    int Q;
    cin >> Q;
    DynamicMas<int> arr;
    for(int i = 0; i < Q; i++) {
        string com;
        cin >> com;
```

```
            if(com == "size") {
                cout << arr.size() << "\n";
            } else if(com == "capacity") {
                cout << arr.cap() << "\n";
            } else if(com == "print") {
                arr.print();
            } else if(com == "get") {
                int num;
                cin >> num;
                cout << arr.get(num) << "\n";
            } else if(com == "set") {
                int a,b;
                cin >> a >> b;
                arr.set(a, b);
            } else if(com == "erase") {
                int a, b;
                cin >> a >> b;
                arr.erase(a, b);
            } else {
                int a, b;
                cin >> a >> b;
                int *ar = new int[b];
                for(int i = 0; i < b; i++)
                    cin >> ar[i];
                arr.insert(a, b, ar);
                delete[] ar;
                ar = nullptr;
            }
        }
        return 0;
    }
```

Результат:

```
PS C:\Users\kutel\.vscode\projects\uni\epic6> g++ algv2.cpp -o algv2 ; if ($?) { .\algv2 }
12

size
0

insert 0 5
251 252 253 254 255

size
5
capacity
8
print
251 252 253 254 255

get 1
252
set 1 777
get 1
777

erase 1 3

get 1
255

size
2
print
251 255
PS C:\Users\kutel\.vscode\projects\uni\epic6>
```

Ця програма була вже на порядок складнішою, тому я написав її за 3,5 години.

Task 6 - Practice# programming: Class Practice Task

Задача полягала в опрацюванні однозв’язного списку та бінарного дерева, використовуючи різні функції. Програма:

```cpp
#include <iostream>
#include <cmath>
using namespace std;

struct Node {
    int value;
    Node* next;
    Node(int val) {
        value = val;
        next = nullptr;
    }
};

struct TreeNode {
    int value;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int v) {
        value = v;
        left = nullptr;
        right = nullptr;
    }
};

class BinaryTree {
    TreeNode *root;
    void mirrorHelper(TreeNode* original, TreeNode*& mirrored) {
        if (original == nullptr) return;
        mirrored = new TreeNode(original->value);
        mirrorHelper(original->right, mirrored->left);
        mirrorHelper(original->left, mirrored->right);
    }
public:
    BinaryTree() {
        root = nullptr;
    }

    void add(int val) {
        if(root == nullptr) {
            root = new TreeNode(val);
        } else {
            TreeNode *cur = root;
                while(true) {
                if(val < cur->value) {
                    if(cur->left == nullptr) {
                        cur->left = new TreeNode(val);
                        break;
                    } else
                        cur = cur->left;
                } else {
                    if(cur->right == nullptr) {
                        cur->right = new TreeNode(val);
                        break;
                    } else
                        cur = cur->right;
                }
            }
            cur = new TreeNode(val);
        }
    }

```

```
    TreeNode* getroot() {
        return root;
    }

    void postorderTraverse(TreeNode* r) {
        if (r != nullptr) {
            postorderTraverse(r->left);
            postorderTraverse(r->right);
            cout << r->value << " ";
        }
    }

    BinaryTree mirror() {
        BinaryTree tree;
        mirrorHelper(root, tree.root);
        return tree;
    }

    void treeSum(TreeNode *r) {
        if(r == nullptr || (r->left == nullptr && r->right == nullptr)) return;
        treeSum(r->left);
        treeSum(r->right);
        r->value = 0;
        if(r->left != nullptr)
            r->value += r->left->value;
        if(r->right != nullptr)
            r->value += r->right->value;
    }
};
```

```
class LinkedList {
    Node* head;
    int size;
public:
    LinkedList() {
        size = 0;
        head = nullptr;
    }

    void pushBack(int value) {
        Node* newNode = new Node(value);
        if(head == nullptr)
            head = newNode;
        else {
            Node* curr = head;
            while(curr->next != nullptr) {
                curr = curr->next;
            }
            curr->next = newNode;
        }
        size++;
    }

    void print() {
        if(head == nullptr)
            cout << "The list is empty";
        else {
            Node* curr = head;
            while(curr != nullptr) {
```

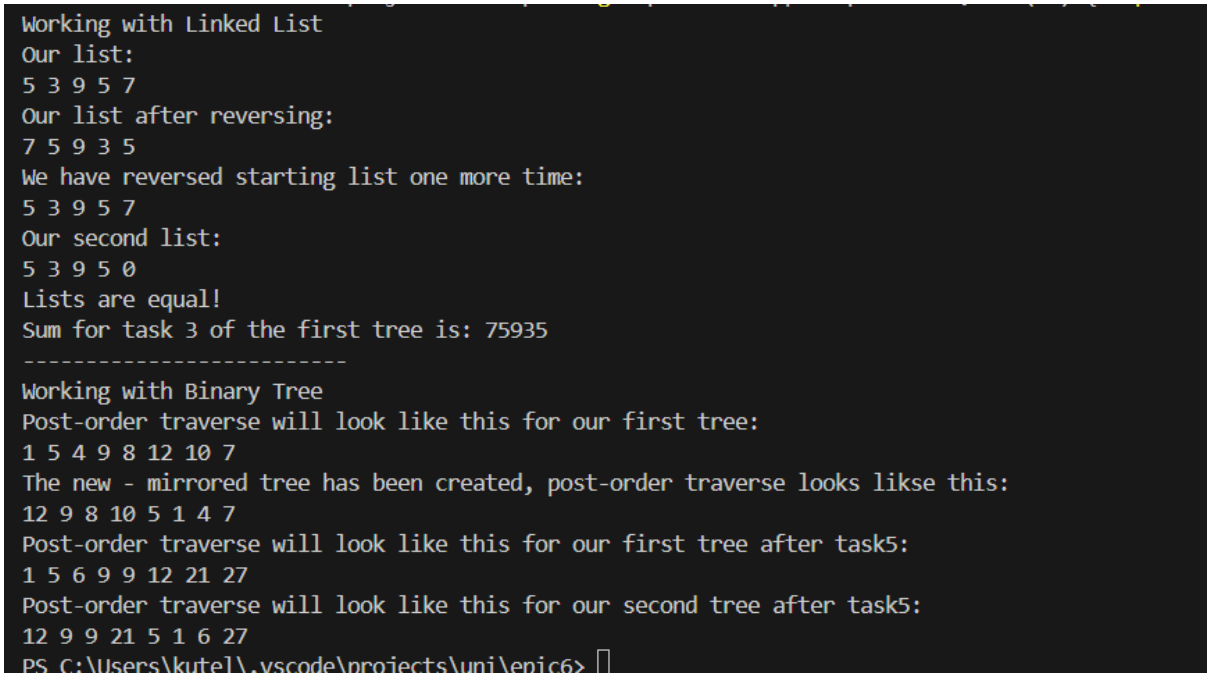
```
        else {
            Node* curr = head;
            while(curr != nullptr) {
                cout << curr->value << " ";
                curr = curr->next;
            }
        }
        cout << endl;
    }

    void reverse() {
        if (!head || !head->next)
            return;
        Node* start = head;
        Node *last = head;
        for(int i = 1; i < size; i++)
            last = last->next;
        while(start != last && start->next != last) {
            int temp = start->value;
            start->value = last->value;
            last->value = temp;

            start = start->next;
            Node* prev = head;
            while(prev->next != last)
                prev = prev->next;
            last = prev;
        }
    }
```

```
148     bool compare(LinkedList l) {
149         if(size != l.size) return false;
150         Node *N1 = head;
151         Node *N2 = l.head;
152         for(int i = 1; i < size; i++) {
153             if(N1->value != N2->value) return false;
154             N1 = N1->next;
155             N2 = N2->next;
156         }
157         return true;
158     }
159 
160     int task3() {
161         Node* newN = head;
162         LinkedList list;
163         int sum = 0;
164         for(int i = 0; i < size; i++) {
165             int n = newN->value * pow(10, i);
166             list.pushBack(n);
167             sum += n;
168             newN = newN->next;
169         }
170         return sum;
171     }
172 };
173 
```

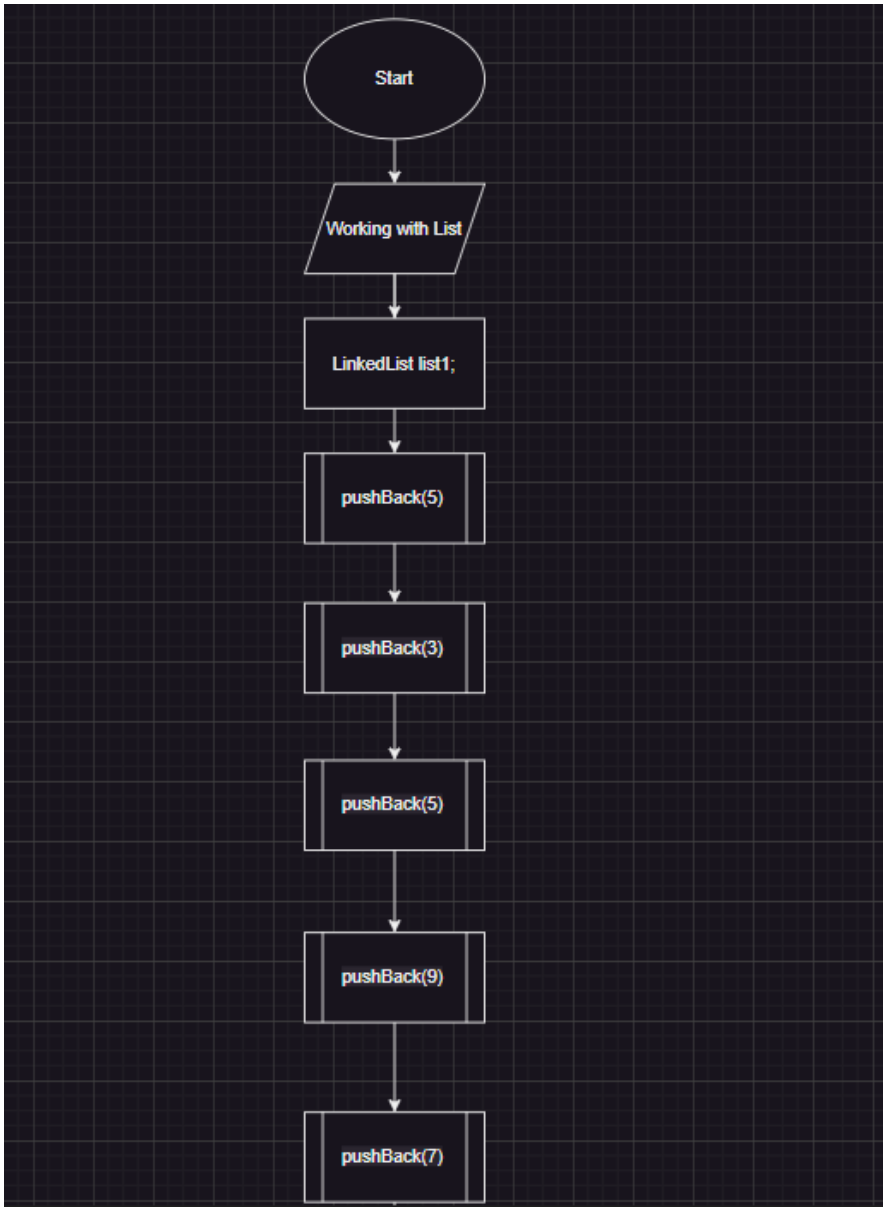
```
174 int main() {
175     cout << "Working with Linked List\n";
176     LinkedList list1;
177     list1.pushBack(5);
178     list1.pushBack(3);
179     list1.pushBack(9);
180     list1.pushBack(5);
181     list1.pushBack(7);
182     cout << "Our list: \n";
183     list1.print();
184     list1.reverse();
185     cout << "Our list after reversing: \n";
186     list1.print();
187     list1.reverse();
188     cout << "We have reversed starting list one more time:\n";
189     list1.print();
190 
191     LinkedList list2;
192     list2.pushBack(5);
193     list2.pushBack(3);
194     list2.pushBack(9);
195     list2.pushBack(5);
196     list2.pushBack(0); // Змінений елмент - 7 щоб були однакові
197     cout << "Our second list:\n";
198     list2.print();
199 
```

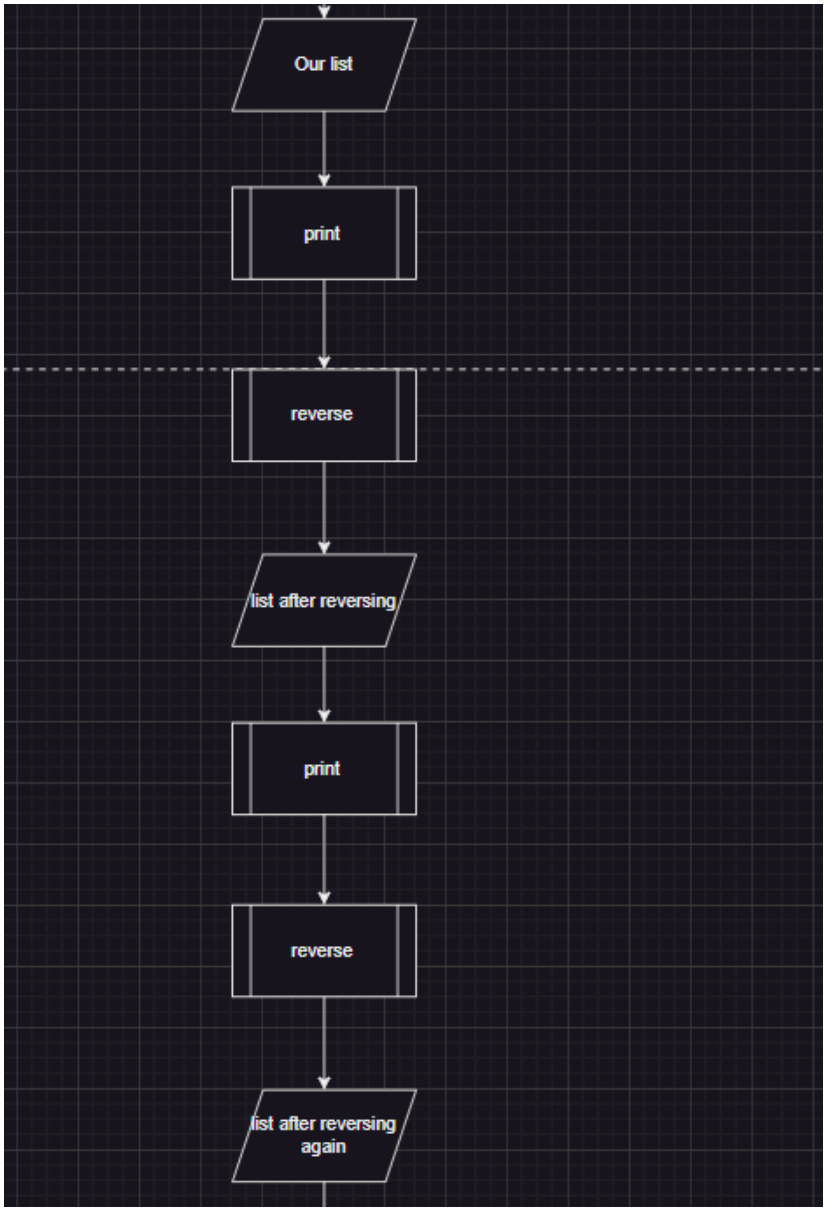
```
    if(list1.compare(list2)) cout << "Lists are equal!\n";
    else cout << "Lists are not equal!\n";

    cout << "Sum for task 3 of the first tree is: " << list1.task3();

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

    cout << "Working with Binary Tree\n";
    BinaryTree tree1;
    tree1.add(7);
    tree1.add(10);
    tree1.add(4);
    tree1.add(1);
    tree1.add(5);
    tree1.add(8);
    tree1.add(9);
    tree1.add(12);
    cout << "Post-order traverse will look like this for our first tree: \n";
    TreeNode* r1 = tree1.getroot();
    tree1.postorderTraverse(r1);
    BinaryTree tree2 = tree1.mirror();
    cout << "\nThe new - mirrored tree has been created, post-order traverse looks likse this:\n";
    TreeNode* r2 = tree2.getroot();
    tree2.postorderTraverse(r2);

    tree2.postorderTraverse(r2);

    tree1.treeSum(r1);
    tree1.treeSum(r2);
    cout << "\nPost-order traverse will look like this for our first tree after task5: \n";
    r1 = tree1.getroot();
    tree1.postorderTraverse(r1);
    cout << "\nPost-order traverse will look like this for our second tree after task5: \n";
    r2 = tree2.getroot();
    tree2.postorderTraverse(r2);
    return 0;
}
```

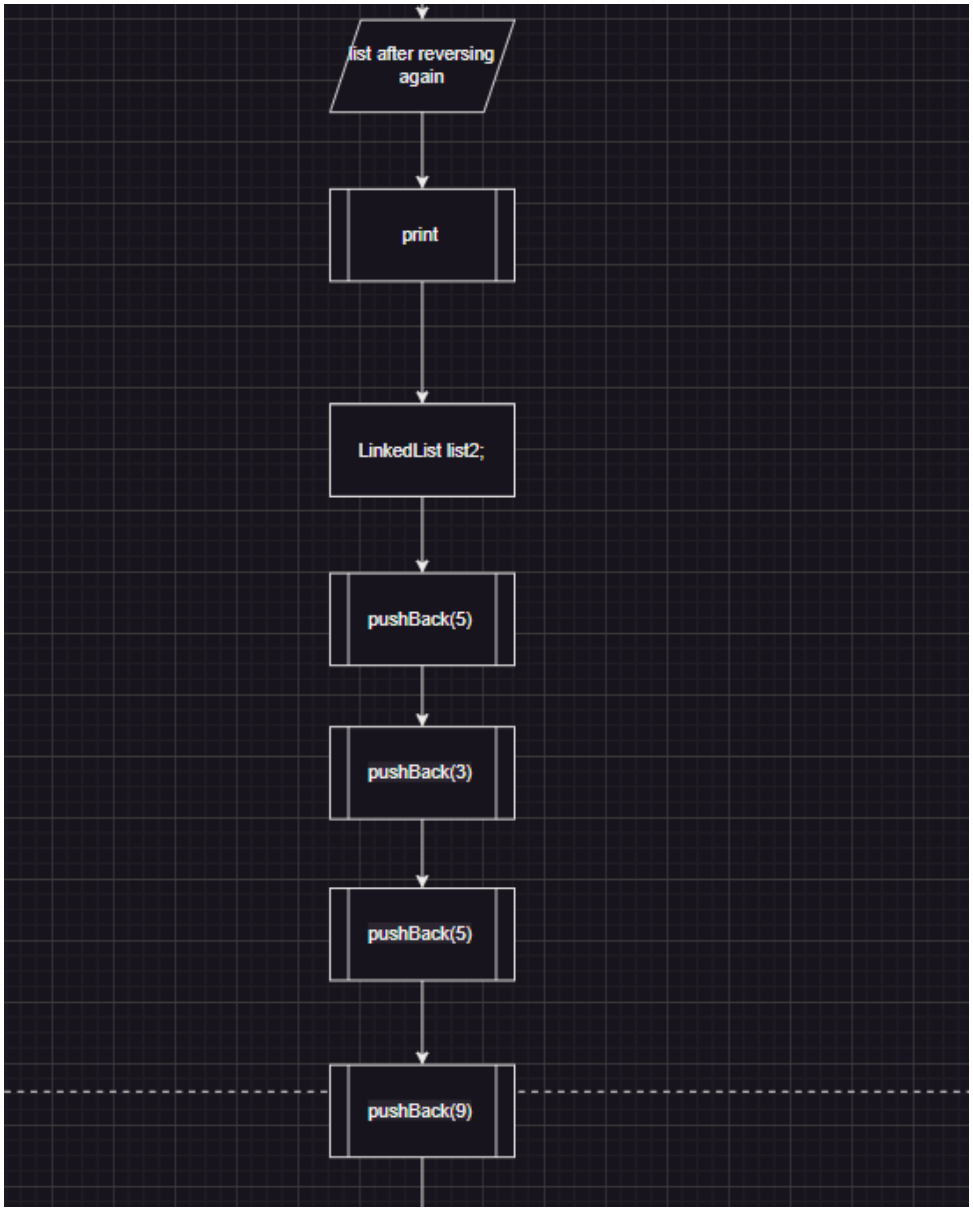
Результат:

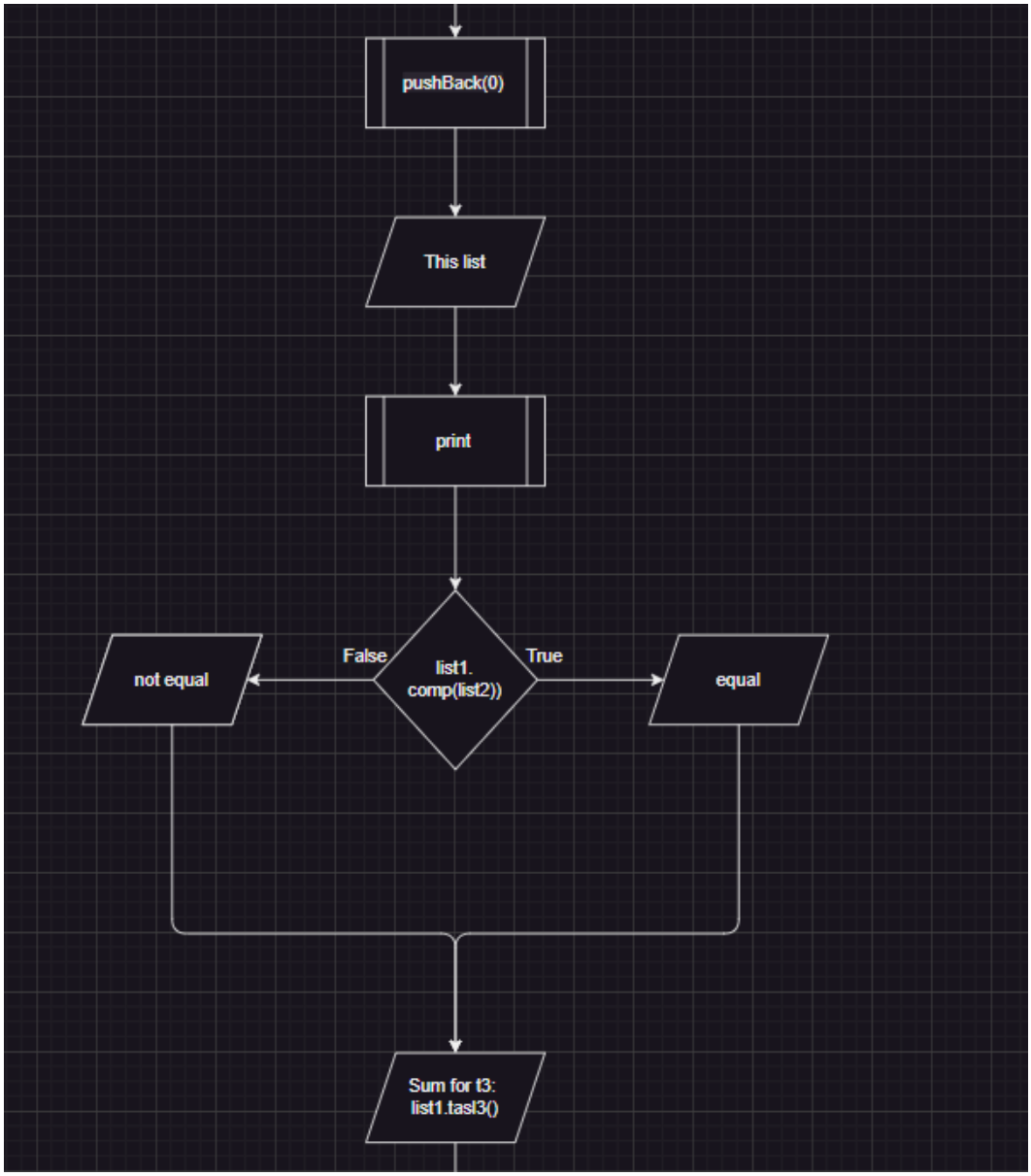


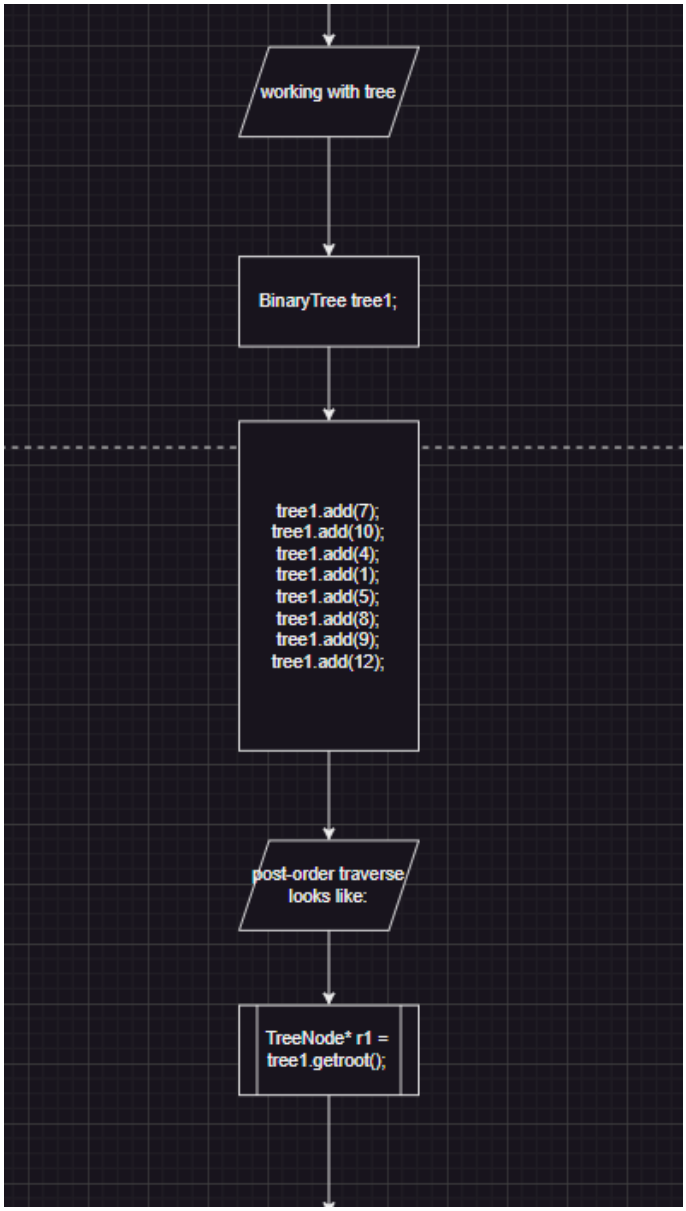
Це найскладніша задача з даного епіку, написав я її за 4 години, блок - схема до неї виглядатиме так:

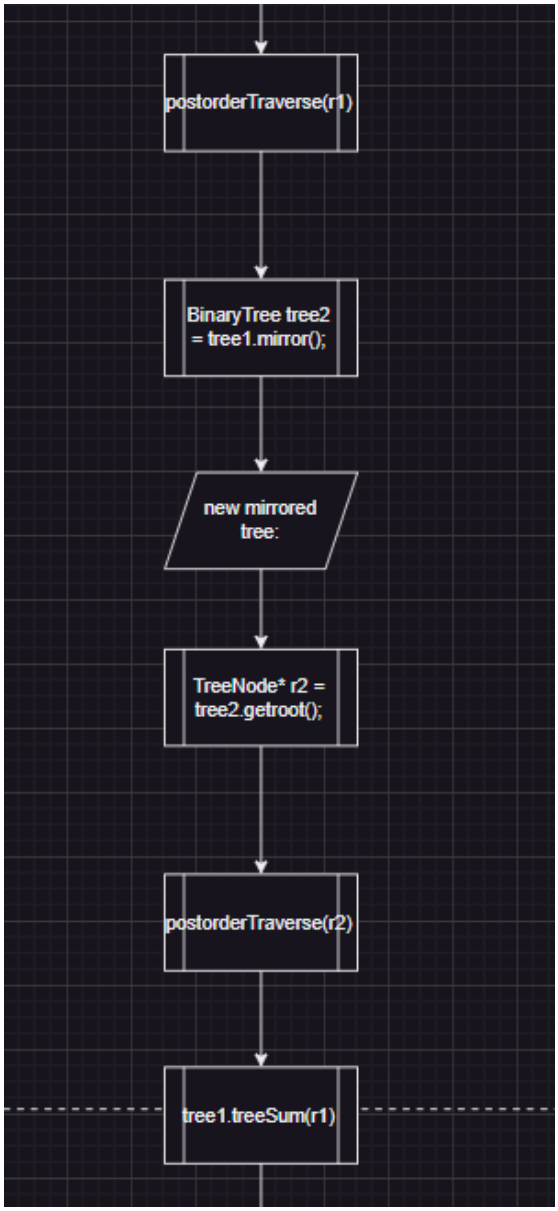

Start
Working with List
LinkedList list1;
pushBack(5)
pushBack(3)
pushBack(5)
pushBack(9)
pushBack(7)

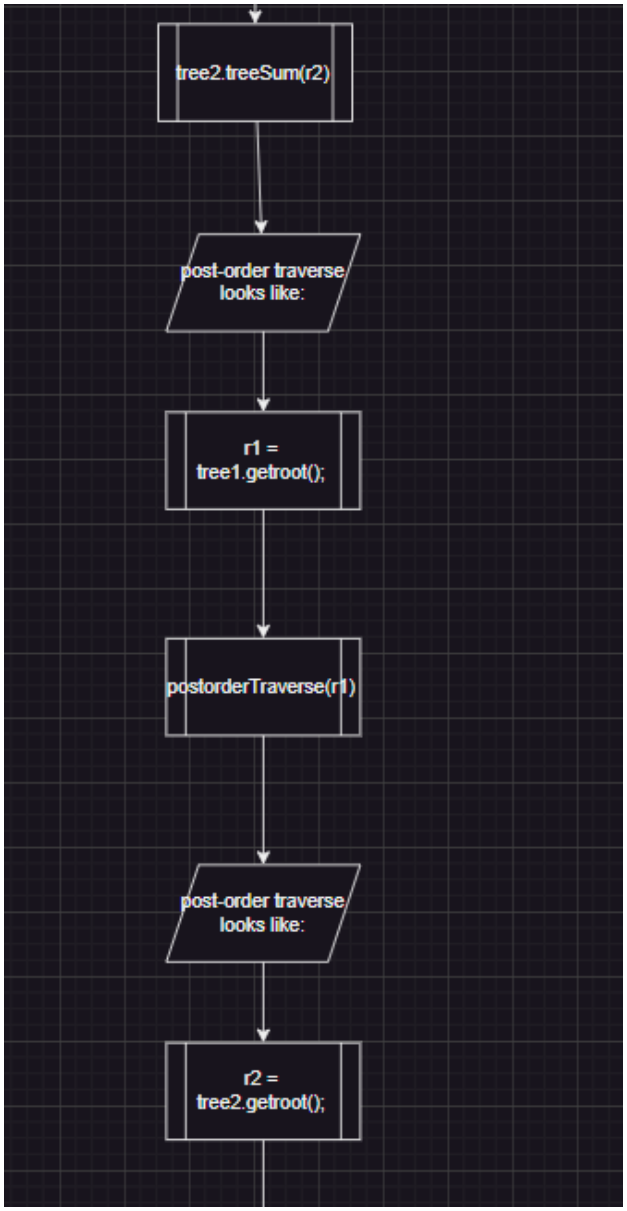

Our list
print
reverse
list after reversing
print
reverse
list after reversing again

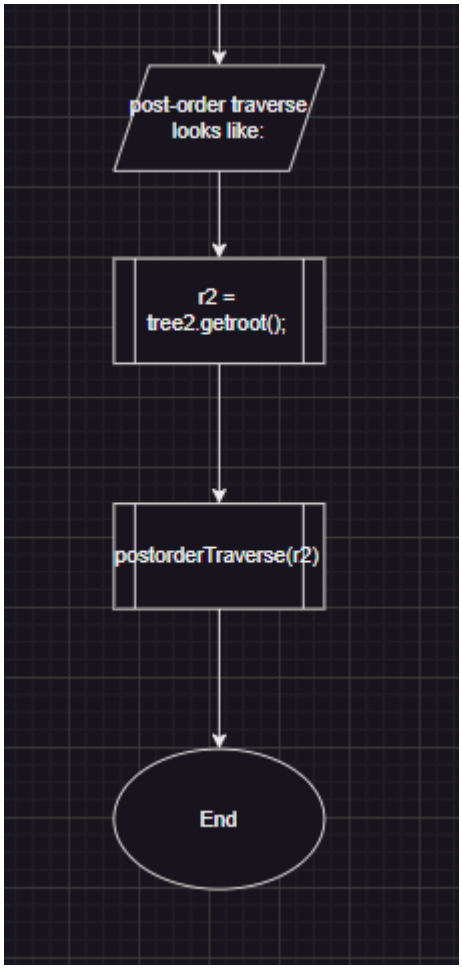

list after reversing again
print
LinkedList list2;
pushBack(5)
pushBack(3)
pushBack(5)
pushBack(9)

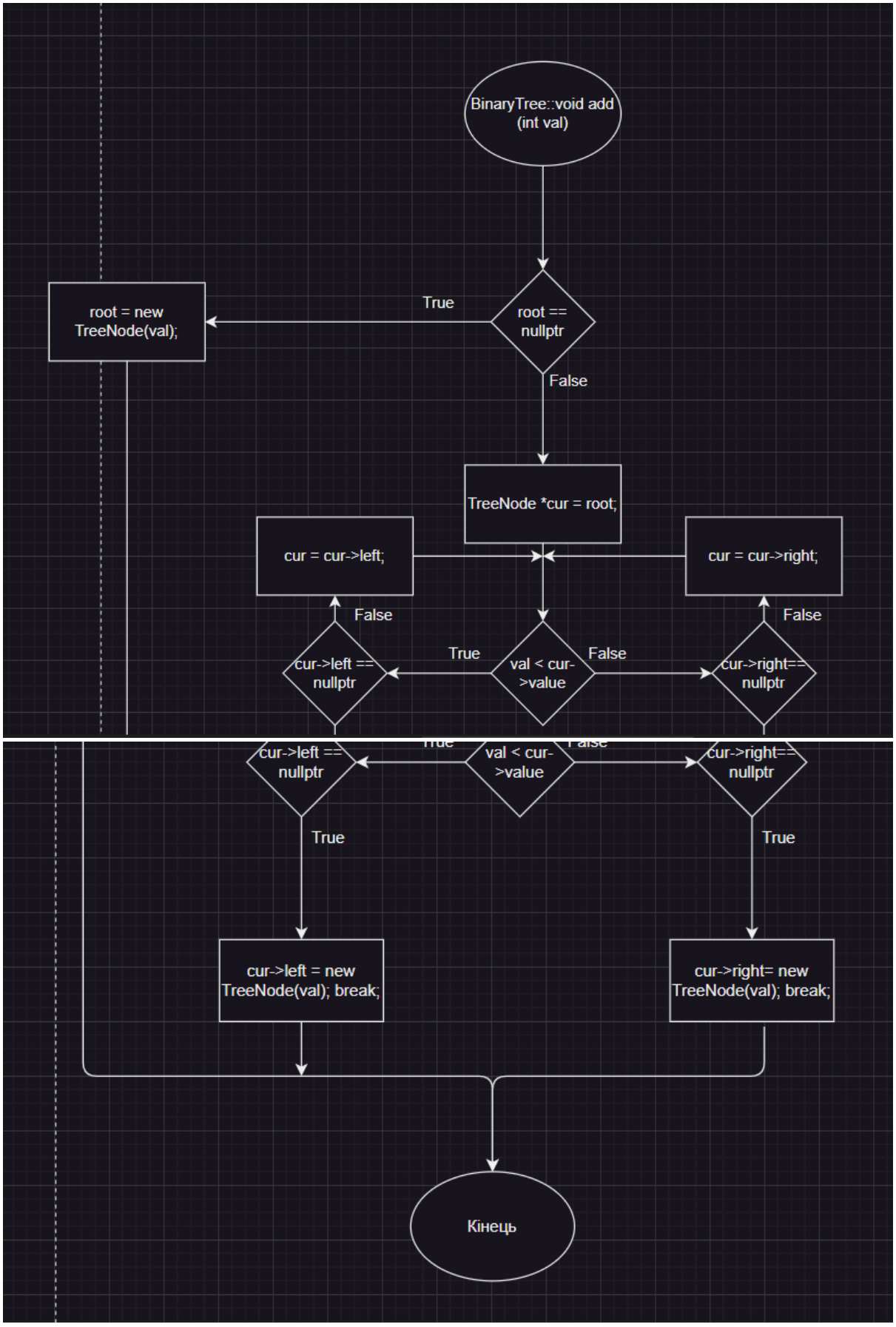

pushBack(0)
This list
print
list1.
comp(list2))
False
True
not equal
equal
Sum for t3:
list1.tasl3()

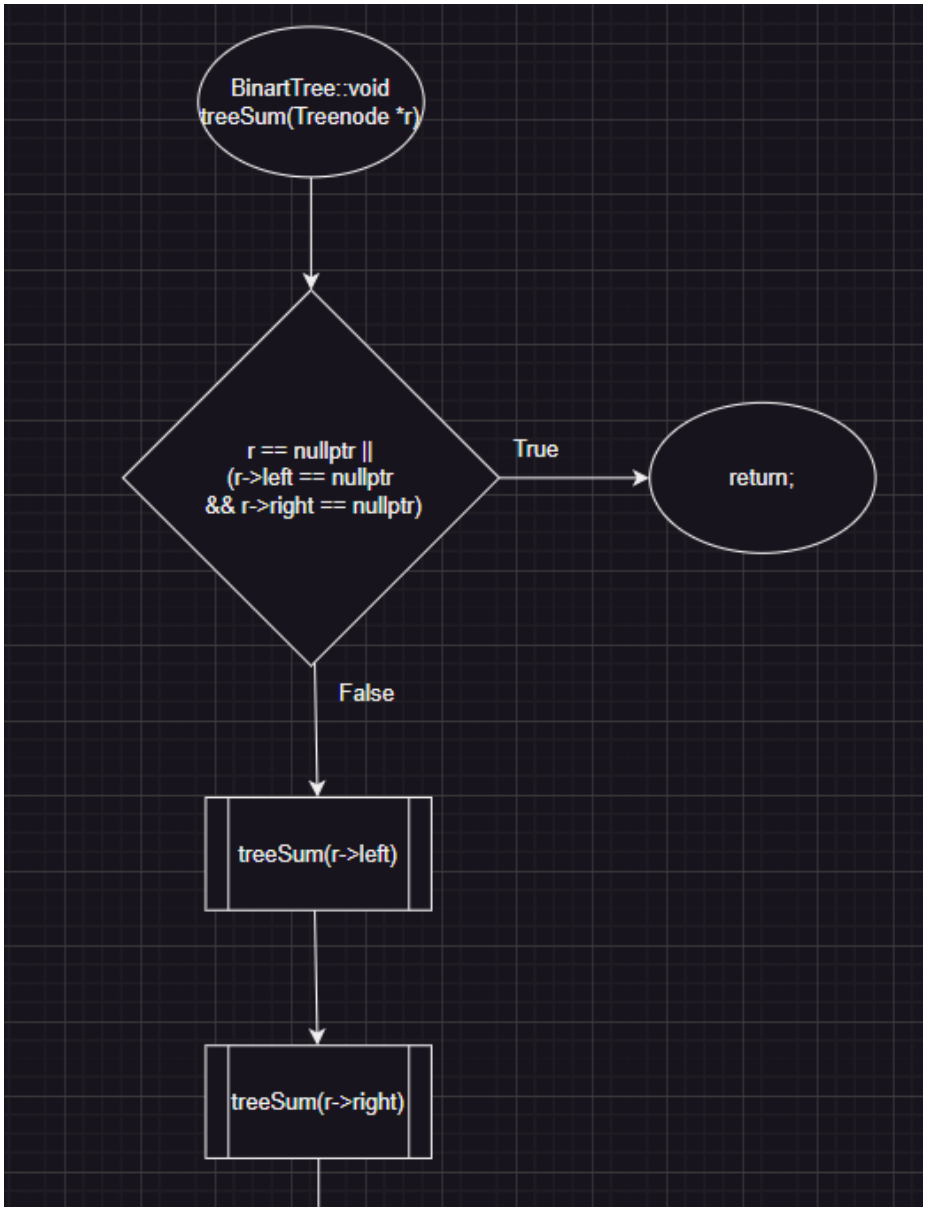

working with tree
BinaryTree tree1;
tree1.add(7);
tree1.add(10);
tree1.add(4);
tree1.add(1);
tree1.add(5);
tree1.add(8);
tree1.add(9);
tree1.add(12);
post-order traverse looks like:
TreeNode* r1 = tree1.getroot();

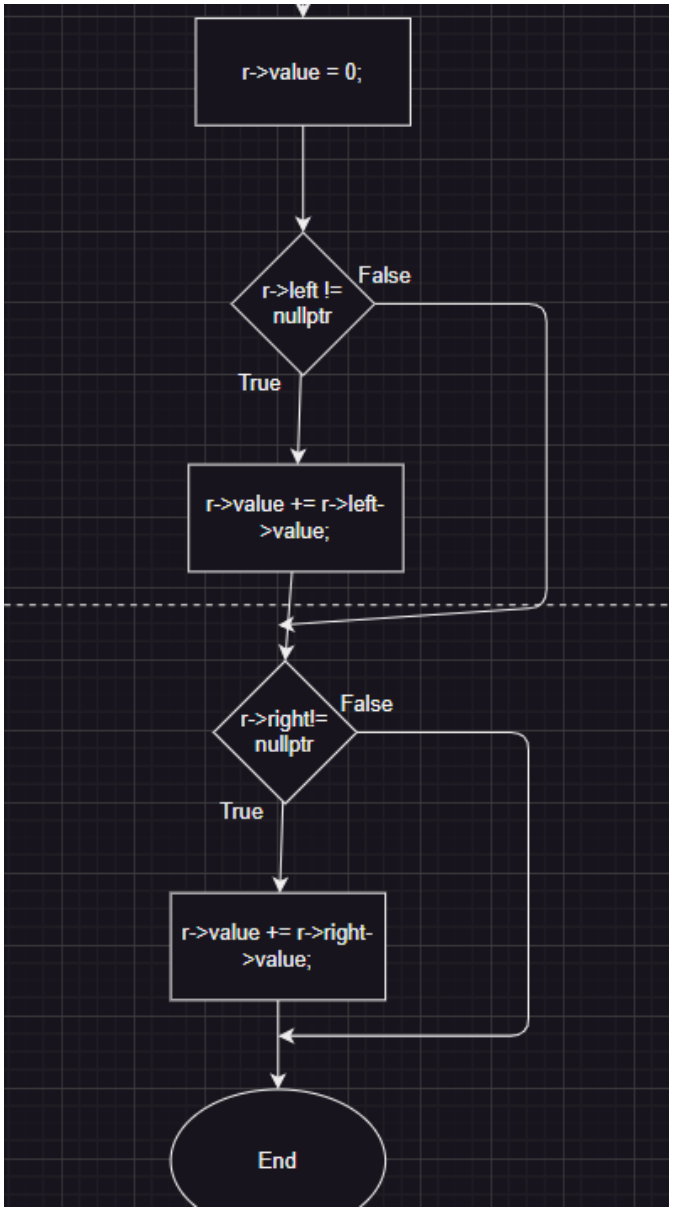

postorderTraverse(r1)
BinaryTree tree2 = tree1.mirror();
new mirrored tree:
TreeNode* r2 = tree2.getroot();
postorderTraverse(r2)
tree1.treeSum(r1)

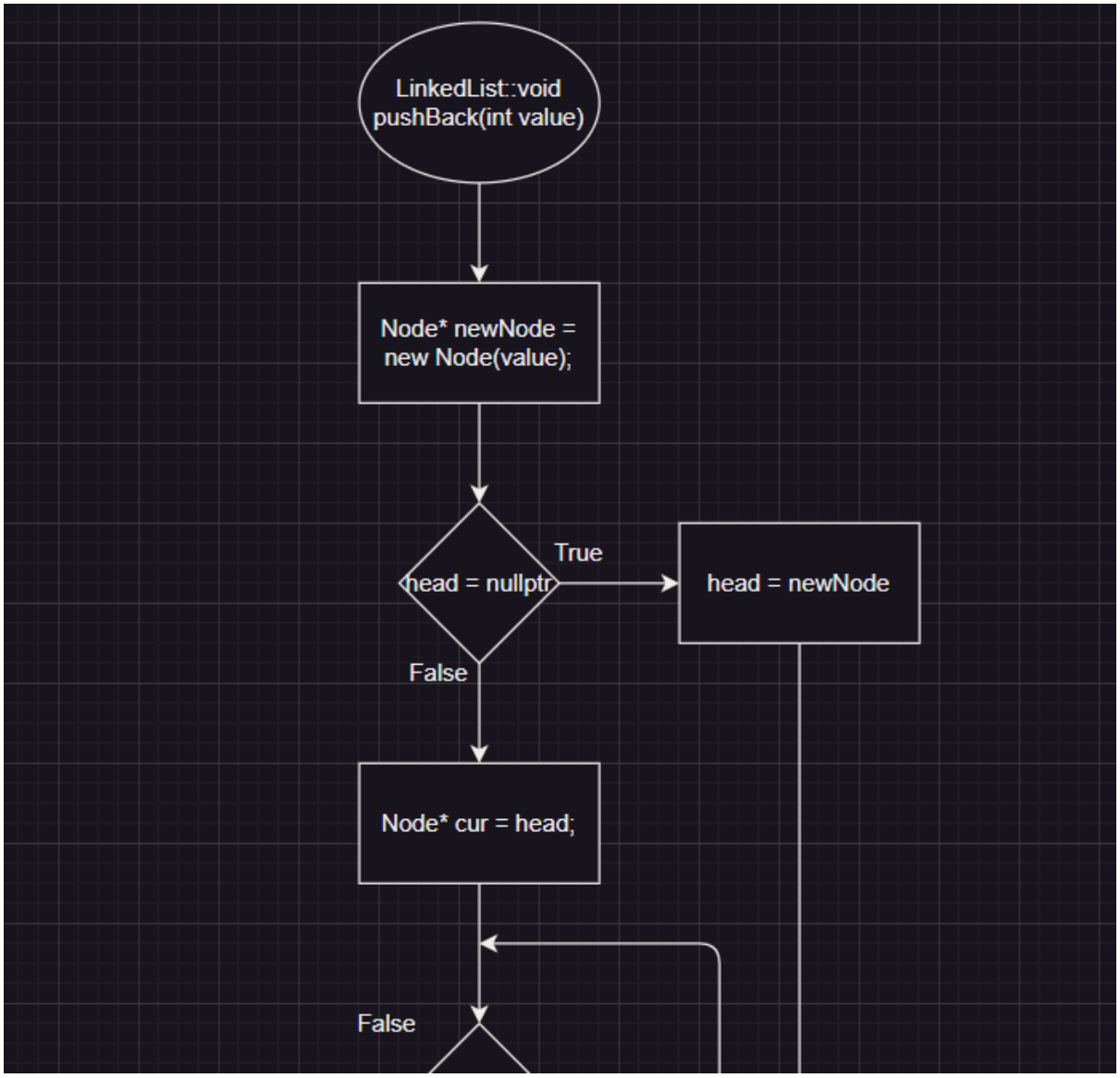

tree2.treeSum(r2)
post-order traverse looks like:
r1 = tree1.getroot();
postorderTraverse(r1)
post-order traverse looks like:
r2 = tree2.getroot();

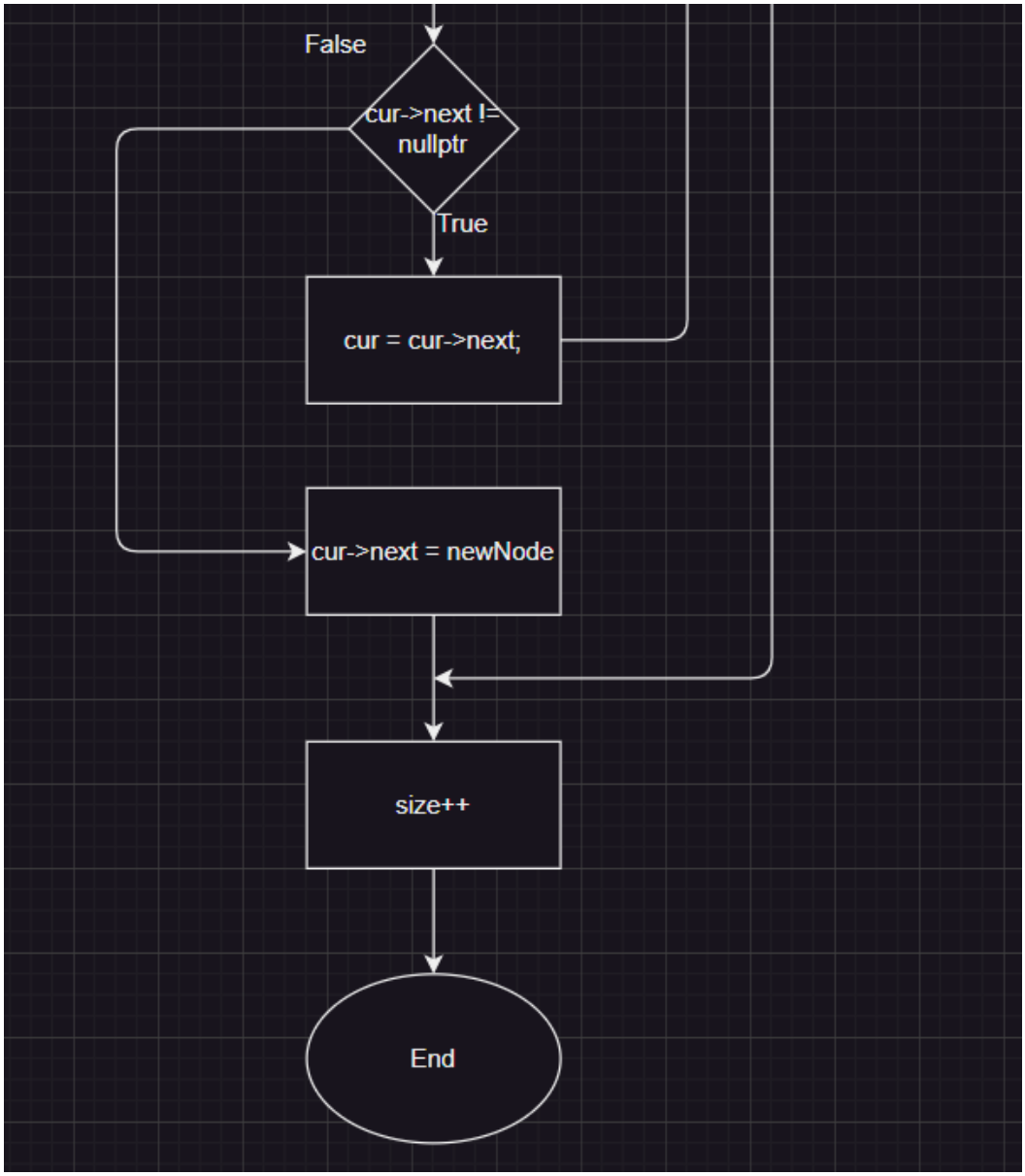

post-order traverse looks like:
r2 = tree2.getroot();
postorderTraverse(r2)
End

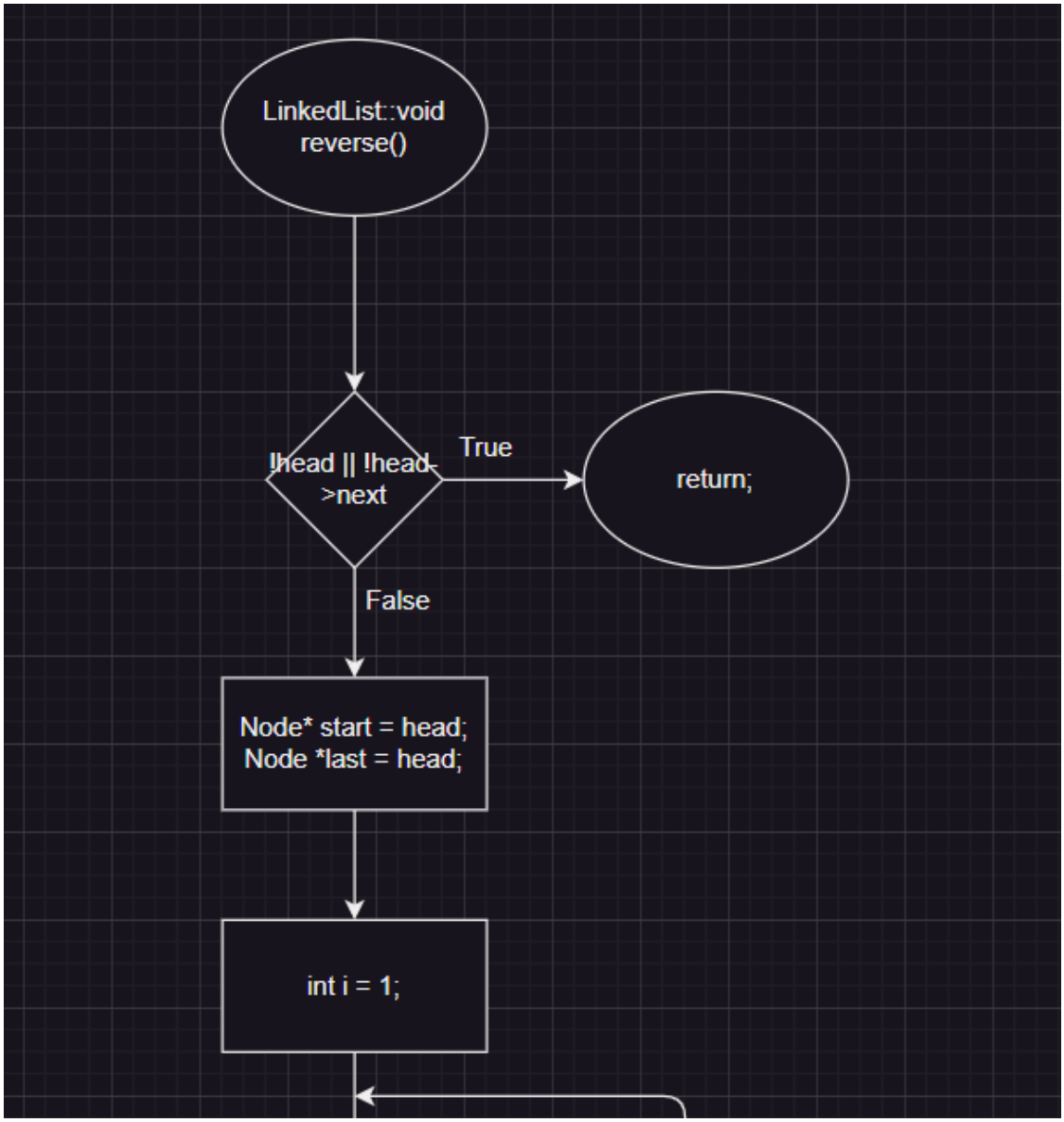

BinaryTree::void add (int val)
root == nullptr
True
root = new TreeNode(val);
False
TreeNode *cur = root;
cur = cur->left;
cur = cur->right;
False
False
cur->left == nullptr
True
val < cur->value
False
cur->right== nullptr
cur->left == nullptr
val < cur->value
cur->right== nullptr
True
True
cur->left = new TreeNode(val); break;
cur->right= new TreeNode(val); break;
Кінець

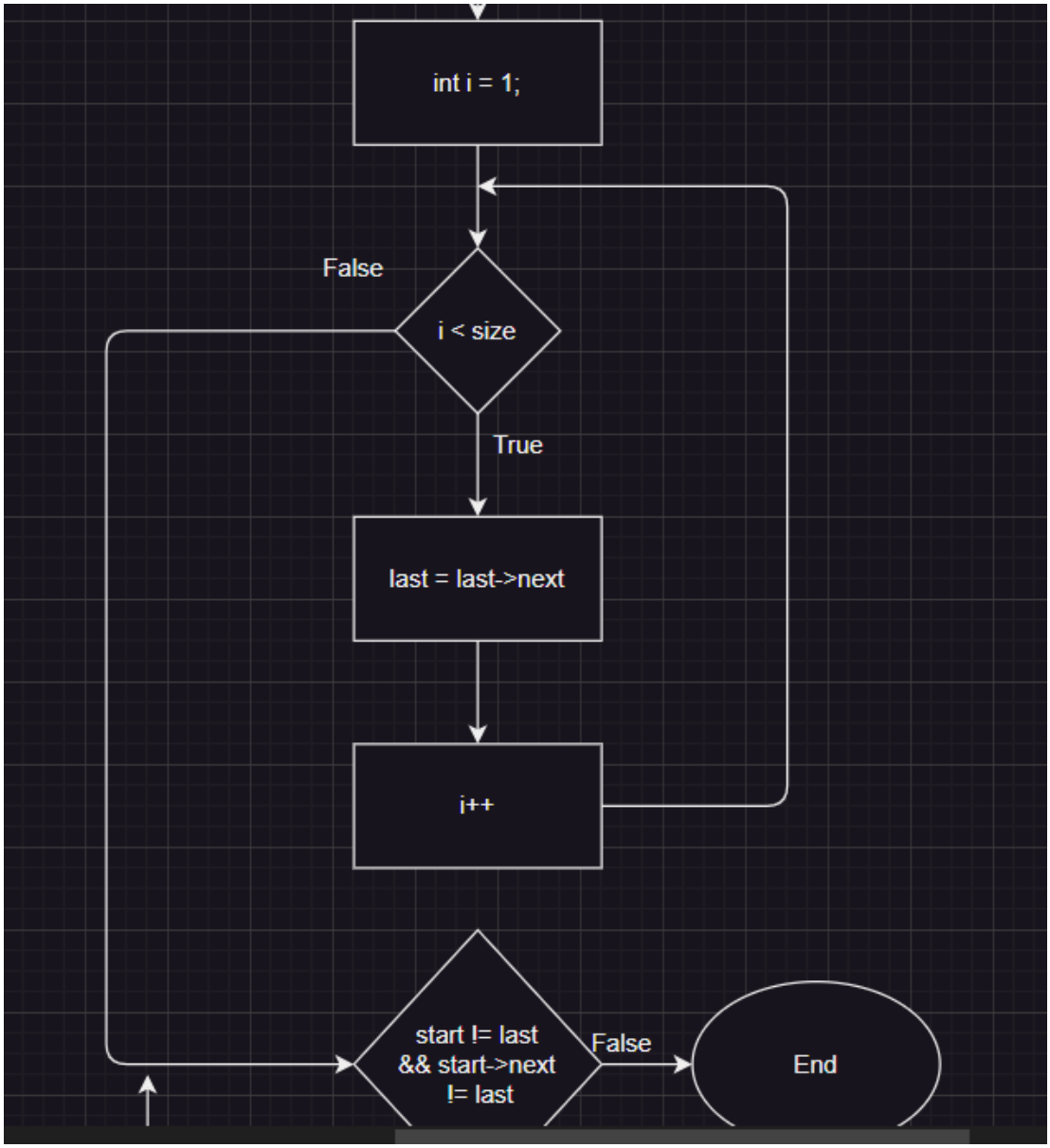

BinartTree::void
treeSum(Treenode *r)
r == nullptr ||
(r->left == nullptr
&& r->right == nullptr)
True
return;
False
treeSum(r->left)
treeSum(r->right)

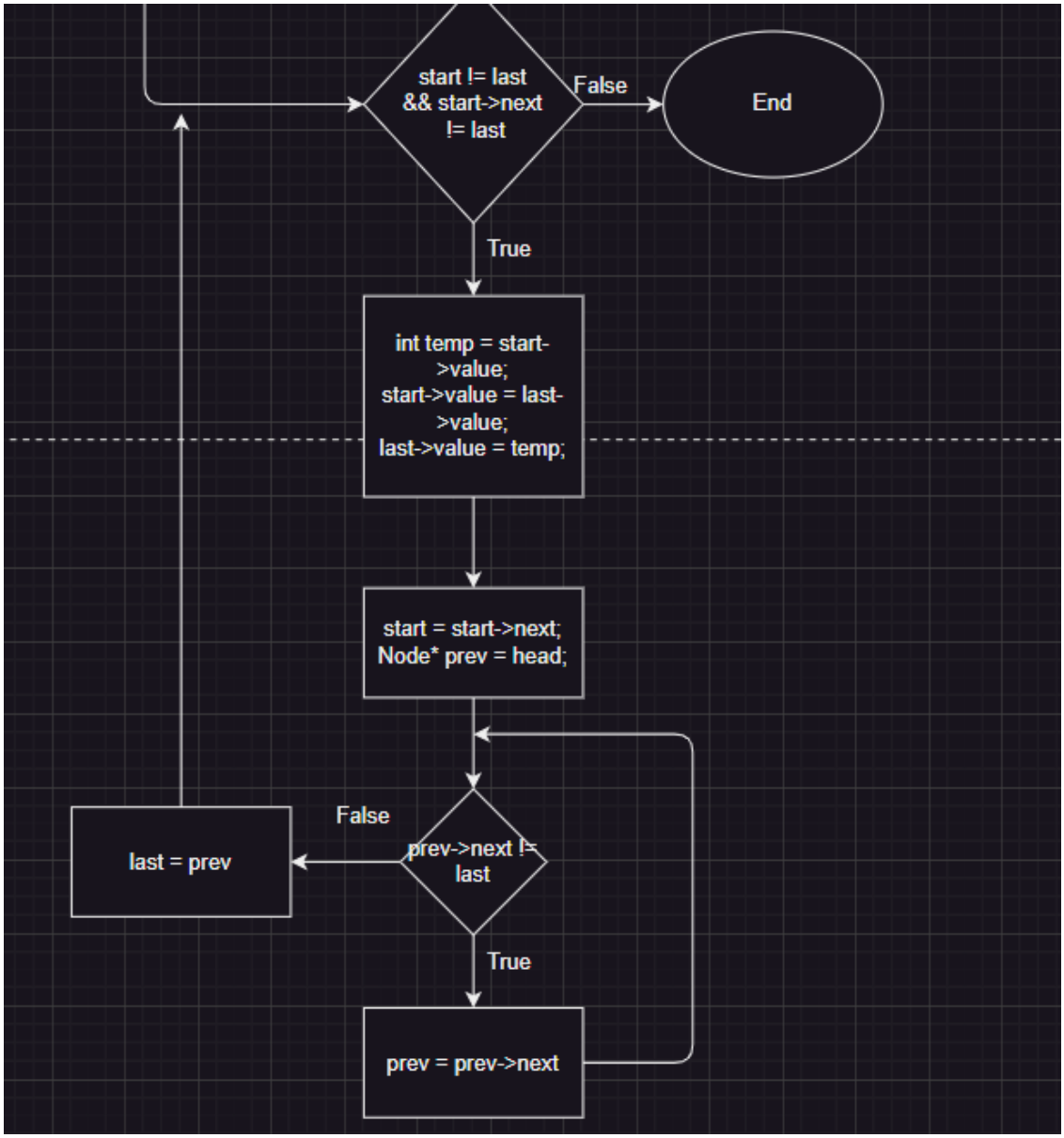

r->value = 0;
r->left != nullptr
False
True
r->value += r->left->value;
r->right!= nullptr
False
True
r->value += r->right->value;
End


LinkedList::void
pushBack(int value)
Node* newNode =
new Node(value);
head = nullptr
True
head = newNode
False
Node* cur = head;
False


False
cur->next !=
nullptr
True
cur = cur->next;
cur->next = newNode
size++
End


LinkedList::void reverse()
!head || !head->next
True
return;
False
Node* start = head;
Node *last = head;
int i = 1;


int i = 1;
False
i < size
True
last = last->next
i++
start != last
&& start->next
!= last
False
End



На написання цієї блок-схеми я витратив 2 години.

Task 7 - Practice# programming: Self Practice Task

https://algotester.com/uk/ArchiveProblem/DisplayWithEditor/40829

Шифрування корупційних схем

Я не зміг знайти задачі, яка б стосувалась епіка напряму, тому обрав цю цікавеньку задачу, з цікавим описом.

Умова:

В офісі ЗЕника працює один татарин. Щомісяця він придумує нову геніальну корупційну схему й охоче ділиться нею із Зеником у повідомленні. Цього місяця знову не обійшлося без нової хитрої схеми. Однак Татарин — не дурний. Він знає, що повідомлення зі схемою для Зеника можуть перехопити правоохоронці. Тому він шифрує його алгоритмом кодування довжин серій.

Кодування довжин серій (англ. Run-length encoding, RLE) — простий алгоритм стиснення даних, який оперує серіями даних, тобто послідовностями, у яких один і той ж символ зустрічається кілька разів поспіль. При кодуванні рядок однакових символів, що становлять серію, замінюють рядком, який містить сам повторюваний символ і кількість його повторів (З Вікіпедії).

Розгляньмо приклад кодування рядка `AAAABBBBBBBACBBBDDDDDDDDDDD`. Якщо застосувати алгоритм RLE до нього, то отримаємо `4A7B1A1C4B11D`. Останній запис інтерпретується як чотири `A`, сім `B`, одна `A`, одна `C`, чотири `B`, одинадцять `D`.

Зверніть увагу, що `4A4B3B1A1C4B7D4D` не є правильно закодованим рядком — треба записувати цілі серії. `4A7BAC4B11D` також не є правильно закодованим рядком — навіть якщо довжина серії символа дорівнює одиниці, то все одно потрібно записувати `1`.

Зашифруйте корупційну схему татарина алгоритмом RLE.

Вхідні дані

Вхідні дані містять єдиний рядок s — корупційну схему татарина.

Вихідні дані

Виведіть закодовану за алгоритмом RLE корупційну схему.

Довжина закодованої схеми не перевищуватиме 10^5 символів.

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.005	1.297	Перегляд
3 хвилини тому	C++ 23	Неправильна відповідь 18	0.003	1.141	Перегляд

Програма:

```cpp
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char s[100001];
    cin >> s;
    char ch = s[0];
    int was = 1;
    int l = strlen(s);
    for(int i = 1; i < l; i++) {
        if(s[i]==ch) {
            was++;
        } else {
            cout << was;
            was = 1;
            cout << ch;
            ch = s[i];
        }
    }
    cout << was << ch;
    return 0;
}
```

Результат:

```
PS C:\Users\kutel\.vscode\projects\uni\epic6
FFFFLLLLLLAFSDLLLLDSIIIDUUVVWWWW
4F6L1A1F1S1D4L1D1S3I1D2U2V4W
PS C:\Users\kutel\.vscode\projects\uni\epic6
```

Загалом задачка не дуже складна, тому я її написав за 15 хв.

Робота в команді

Загалом, працюючи в команді, ми обговорили як реалізувати задані нам динамічні структури та допомагали одне одному з труднощами. Ось один із скріншотів наших зустрічей:

Висновок: Я отримав практичні навички у роботі з динамічними структурами, алгоритмами обробок динамічних структур