

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту



Звіт

про виконання лабораторних та практичних робіт блоку № 6

На тему: «Програмування: алгоритм, програма, код. Системи числення. Двійкова система числення. Розробка та середовище розробки програми.»

з дисципліни: «Основи програмування»

до:

Практичних Робіт до блоку № 6

Виконала:

Студентка групи ШІ-13
Бобринок Ангеліна Вадимівна

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

Мета: навчитись працювати з динамічними структурами, реалізувати зв'язний список та навчитись працювати з бінарними деревами.

Теоретичні відомості:

1. Основи Динамічних Структур Даних:
 - Вступ до динамічних структур даних: визначення та важливість
 - Виділення пам'яті для структур даних (stack і heap)
2. Стек:
 - Визначення та властивості стеку
 - Операції push, pop, top: реалізація та використання
 - Приклади використання стеку: обернений польський запис, перевірка балансу дужок
 - Переповнення стеку
3. Черга:
 - Визначення та властивості черги
 - Операції enqueue, dequeue, front: реалізація та застосування
4. 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
5. Дерева:
 - Бінарні дерева: вставка, пошук, видалення
 - Обхід дерева: в глибину (preorder, inorder, postorder), в ширину
 - Застосування дерев: дерева рішень, хеш-таблиці
 - Складніші приклади дерев: AVL, Червоно-чорне дерево
6. 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

Написати функцію для створення списку. Функція може створювати порожній список, а потім додавати в нього елементи.

2. Написати функцію для друку списку. Функція повинна передбачати вивід повідомлення, якщо список порожній.

3. Написати функції для знищення й додавання елементів списку у відповідності зі своїм варіантом.

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

5. Написати функцію для запису списку у файл.

6. Написати функцію для знищення списку.

7. Записати список у файл, знищити його й виконати друк (при друці повинне бути видане повідомлення "Список порожній").

8. Написати функцію для відновлення списку з файлу.

9. Відновити список і роздрукувати його.

10. Знищити список.

Записи в лінійному списку містять ключове поле типу `int`. Сформувати однонаправлений список. Знищити з нього `K` елементів, починаючи із заданого номера, додати елемент перед елементом із заданим ключем;

Завдання №2

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 №3

Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".

Ви отримаєте `QQ` запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.

Вам будуть поступати запити такого типу:

- **Вставка:**

Ідентифікатор - `insertinsert`

Ви отримуєте ціле число `indexindex` елемента, на місце якого робити вставку.

Після цього в наступному рядку рядку написане число `NN` - розмір масиву, який треба вставити.

У третьому рядку `NN` цілих чисел - масив, який треба вставити на позицію `indexindex`.

- **Видалення:**

Ідентифікатор - `eraseerase`

Ви отримуєте 2 цілих числа - `indexindex`, індекс елемента, з якого почати видалення та `nn` - 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.

Ви виводите кількість зарезервованої пам'яті у динамічному масиві.

Ваша реалізація динамічного масиву має мати фактор росту (Growth factor) рівний 2.

- **Отримання значення ii-го елемента**

Ідентифікатор - `get`

Ви отримуєте ціле число - `index`, індекс елемента.

Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора `[]`

- **Модифікація значення ii-го елемента**

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 `<<`

Завдання №4

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

Умови задачі:

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

`bool compare(Node *h1, Node *h2);`

Умови задачі:

- 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*.

`Node* add(Node *n1, Node *n2);`

Умови задачі:

- 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 списку (напр. 379 \Rightarrow 9 \rightarrow 7 \rightarrow 3);
- функція повертає новий список, передані в функцію списки не модифікуються.

`TreeNode *create_mirror_flip(TreeNode *root);`

Умови задачі:

- використовувати цілі числа для значень у вузлах дерева
- реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву гілки дерева

- функція повертає нове дерево, передане в функцію дерево не модифікується
void tree_sum(TreeNode *root);

Умови задачі:

- 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

Завдання №5

Снігурочка — на сьогодні дуже велика рідкість, одним словом, вимираючий вид. А от тих Дідів Морозів порозводилося вже. Хоч бери й відстрілюй...

І ось напередодні Нового року снігурочка вирішила скористатися своїм монополістичним становищем і оголосила конкурс для Дідів Морозів. Переможець конкурсу стане якраз тим щасливчиком, якому вона складе пару в новорічну ніч.

Правила конкурсу до болю прості. Кожен Дід Мороз приходить і урочисто зачитує снігурочці свій вірш, який він готував протягом довгих недоспаних ночей.

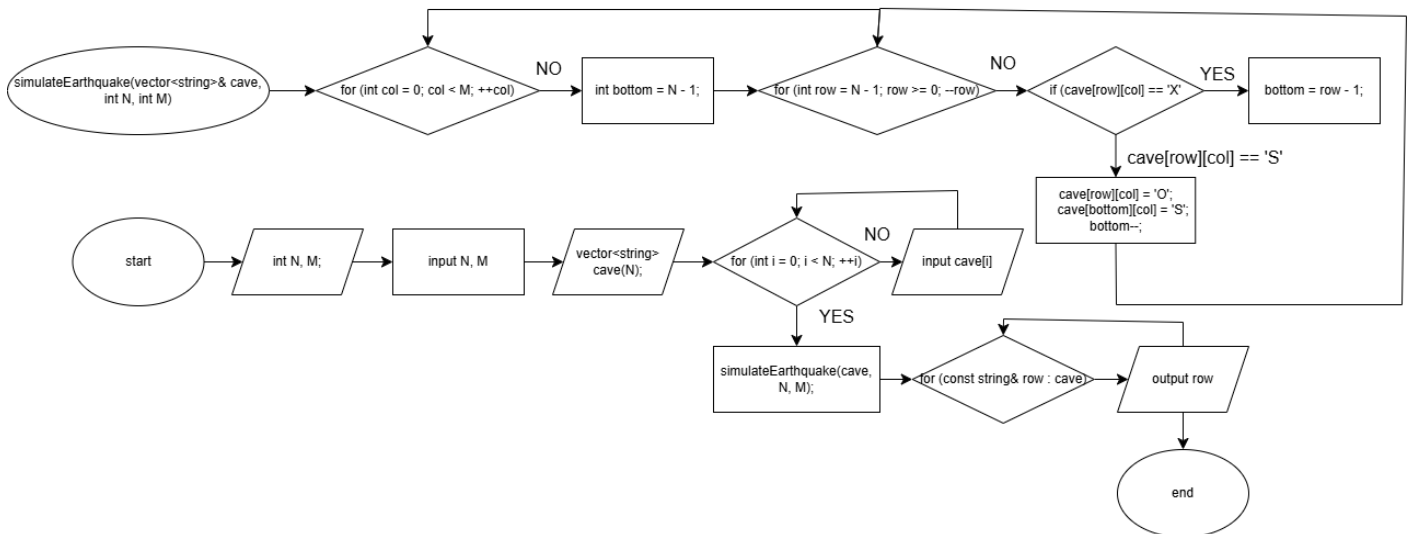
Снігурочка ж у свою чергу ставить кожному з них оцінку — ціле невід’ємне число. Стратегія оцінювання снігурочки теж до болю тупа — вона підраховує кількість паліндромів у вірші дідуся-претендента і трактує її як його оцінку.

Нагадаю, що паліндром — це слово, що читається однаково як зліва направо, так і справа наліво. При цьому регістр букв не враховується.

Кількість кандидатів на новорічну ніч є величезна, от і вирішила Снігурочка автоматизувати процес оцінювання.

Вам і доведеться написати програму, яка оцінює вірш невідомого Діда Мороза.


2) Дизайн та планована оцінка часу виконання завдань:






Завдання №1

3) Конфігурація середовища до виконання завдань:

Планування роботи в notion



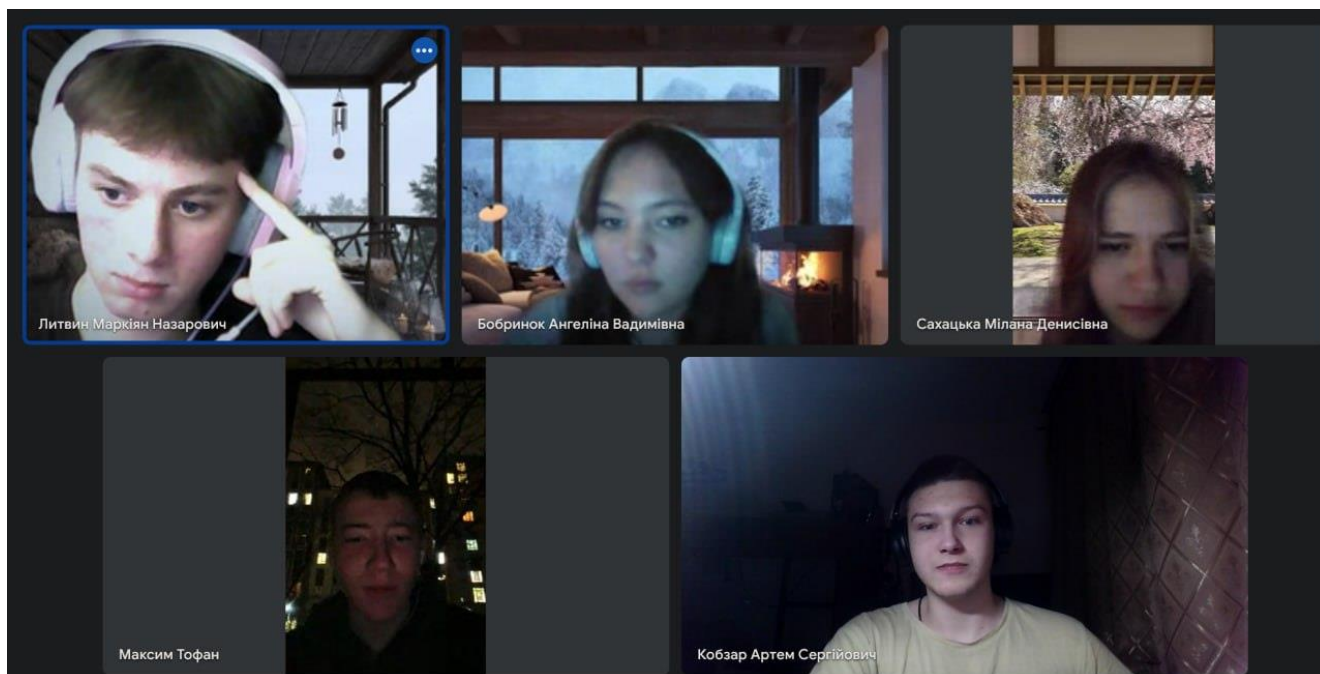
epic 6

	<div></> Task</div>	<div>  <div>Status</div> </div>	<div>  <div>deadline</div> </div>
Anhelina	Theory Education Activities	<div> <div></div> <div>Done</div> </div>	05/12/2024
Anhelina	Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-9)	<div> <div></div> <div>Pending</div> </div>	05/12/2024
Anhelina	Lab# programming: VNS Lab 10	<div> <div></div> <div>In progress</div> </div>	05/12/2024
Anhelina	Lab# programming: Algotester Lab 5	<div> <div></div> <div>Pending</div> </div>	05/12/2024
Anhelina	Lab# programming: Algotester Lab 7-8	<div> <div></div> <div>Pending</div> </div>	05/12/2024
Anhelina	Practice# programming: Class Practice Task	<div> <div></div> <div>Pending</div> </div>	05/12/2024
Anhelina	Practice# programming: Self Practice Task	<div> <div></div> <div>Pending</div> </div>	05/12/2024
Anhelina	Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)	<div> <div></div> <div>Pending</div> </div>	05/12/2024
Anhelina	Results Evaluation and Release	<div> <div></div> <div>Not started</div> </div>	05/12/2024

+

New page

Зустріч з командою та обговорення питань



4) Код програм з посиланням на зовнішні ресурси:

Завдання №1

vns_lab_10_anhelina_bobrynok.cpp

```
1  #include <iostream>
2  #include <fstream>
3
4  struct Node {
5      int key;
6      Node* next;
7
8      Node(int k) : key(k), next(nullptr) {}
9  };
10
11 class LinkedList {
12 private:
13     Node* head;
14
15 public:
16     LinkedList() : head(nullptr) {}
17
18     // Функція перевірки, чи список порожній
19     bool isEmpty() const {
20         return head == nullptr;
21     }
22
23     // Функція для додавання елемента в кінець списку
24     void add(int key) {
25         Node* newNode = new Node(key);
26         if (isEmpty()) {
27             head = newNode;
28         } else {
29             Node* current = head;
30             while (current->next) {
31                 current = current->next;
32             }
```

```

33         current->next = newNode;
34     }
35 }
36
37 // Функція для друку списку
38 void print() const {
39     if (isEmpty()) {
40         std::cout << "Список порожній.\n";
41         return;
42     }
43     Node* current = head;
44     while (current) {
45         std::cout << current->key << " ";
46         current = current->next;
47     }
48     std::cout << "\n";
49 }
50
51 // Функція для видалення K елементів, починаючи з заданого номера
52 void deleteKElementsFrom(int startPos, int k) {
53     if (isEmpty()) {
54         std::cout << "Список порожній. Видалення неможливе.\n";
55         return;
56     }
57
58     if (startPos < 1) {
59         std::cout << "Номер позиції повинен бути >= 1.\n";
60         return;

```



```
57
58     if (startPos < 1) {
59         std::cout << "Номер позиції повинен бути >= 1.\n";
60         return;
61     }
62
63     Node* current = head;
64     Node* prev = nullptr;
65
66     for (int i = 1; i < startPos && current; ++i) {
67         prev = current;
68         current = current->next;
69     }
70
71     for (int i = 0; i < k && current; ++i) {
72         Node* temp = current;
73         current = current->next;
74         delete temp;
75     }
76
77     if (prev) {
78         prev->next = current;
79     } else {
80         head = current;
81     }
82 }
83
```

```

84 // Функція для додавання елемента перед вузлом із заданим ключем
85 void addBefore(int key, int newKey) {
86     Node* newNode = new Node(newKey);
87     if (isEmpty()) {
88         std::cout << "Список порожній. Операція неможлива.\n";
89         delete newNode;
90         return;
91     }
92
93     if (head->key == key) {
94         newNode->next = head;
95         head = newNode;
96         return;
97     }
98
99     Node* current = head;
100     Node* prev = nullptr;
101
102     while (current && current->key != key) {
103         prev = current;
104         current = current->next;
105     }
106
107     if (current) {
108         newNode->next = current;
109         if (prev) {
110             prev->next = newNode;
111         }
112     } else {
113         std::cout << "Елемент із ключем " << key << " не знайдено.\n";
114         delete newNode;
115     }
116 }
117
118 // Функція для запису списку у файл
119 void writeToFile(const std::string& filename) const {
120     std::ofstream file(filename);
121     if (!file) {
122         std::cout << "Помилка запису у файл.\n";
123         return;
124     }
125     Node* current = head;
126     while (current) {
127         file << current->key << " ";
128         current = current->next;
129     }
130     file.close();
131 }
132
133 // Функція для відновлення списку з файлу
134 void readFromFile(const std::string& filename) {
135     std::ifstream file(filename);
136     if (!file) {
137         std::cout << "Помилка читання з файлу.\n";
138         return;
139     }

```

```
141         clear();
142         int key;
143         while (file >> key) {
144             add(key);
145         }
146         file.close();
147     }
148
149     // Функція для очищення списку
150     void clear() {
151         while (head) {
152             Node* temp = head;
153             head = head->next;
154             delete temp;
155         }
156     }
157
158     ~LinkedList() {
159         clear();
160     }
161 };
162
163 int main() {
164     LinkedList list;
165
166     // Додавання елементів
167     list.add(10);
168     list.add(20);
```

```
169     list.add(30);
170     list.add(40);
171     list.add(50);
172     list.print();
173
174     // Видалення K елементів із заданої позиції
175     list.deleteKElementsFrom(2, 2);
176     list.print();
177
178     // Додавання елементу перед заданим ключем
179     list.addBefore(40, 25);
180     list.print();
181
182     // Запис у файл
183     list.writeToFile("list.txt");
184
185     // Видалення списку
186     list.clear();
187     list.print();
188
189     // Відновлення списку з файлу
190     list.readFromFile("list.txt");
191     list.print();
192
193     // Знищення списку
194     list.clear();
195     list.print();
196
197     return 0;
```

Завдання №2

algotester_lab_5_anhelina_bobrynok .cpp

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4
5  using namespace std;
6
7  void simulateEarthquake(vector<string>& cave, int N, int M) {
8      for (int col = 0; col < M; ++col) {
9          int bottom = N - 1; // Починаємо із дна стовпця
10
11         // Проходимо знизу вгору по стовпцю
12         for (int row = N - 1; row >= 0; --row) {
13             if (cave[row][col] == 'X') {
14                 // Якщо зустріли камінь, оновлюємо нове "дно" над каменем
15                 bottom = row - 1;
16             } else if (cave[row][col] == 'S') {
17                 // Переміщуємо пісок на найнижчу доступну клітинку
18                 cave[row][col] = 'O';
19                 cave[bottom][col] = 'S';
20                 bottom--;
21             }
22         }
23     }
24 }
25
26 int main() {
27     int N, M;
28     cin >> N >> M;
29
30     vector<string> cave(N);
31
32     // Зчитування вхідних даних
33     for (int i = 0; i < N; ++i) {
34         cin >> cave[i];
35     }
36
37     // Симуляція землетрусу
38     simulateEarthquake(cave, N, M);
39
40     // Виведення результату
41     for (const string& row : cave) {
42         cout << row << endl;
43     }
44
45     return 0;
46 }
47
48

```

Завдання №3

algotester_lab_78_task_1_anhelina_bobrynok

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  template <typename T>
7  struct DynamicArray {
8      T *arr;
9      int size;
10     int capacity;
11     DynamicArray() {
12         size = 0;
13         capacity = 1;
14         arr = new T[capacity];
15     }
16     ~DynamicArray() {
17         delete[] arr;
18     }
19 };
20
21 template <typename T>
22 void insert(DynamicArray<T> &array, int index, int amount, T *toInsert) {
23     while (array.size + amount >= array.capacity) {
24         array.capacity *= 2;
25     }
26     T *temp = new T[array.capacity];
27     for (int i = 0; i < index; i++) {
28         temp[i] = array.arr[i];
29     }
30     for (int i = 0; i < amount; i++) {
31         temp[index + i] = toInsert[i];
32
33         for (int i = index; i < array.size; i++) {
34             temp[i + amount] = array.arr[i];
35         }
36         array.size += amount;
37         delete[] array.arr;
38         array.arr = temp;
39     }
40
41     template <typename T>
42     void erase(DynamicArray<T> &array, int index, int amount) {
43         T *temp = new T[array.capacity];
44         int acc = 0;
45
46         for (int i = 0; i < array.size; i++) {
47             if (i < index || i >= index + amount) {
48                 temp[acc] = array.arr[i];
49                 acc++;
50             }
51         }
52         array.size -= amount;
53         delete[] array.arr;
54         array.arr = temp;
55     }
56
57     template <typename T>
58     T get(const DynamicArray<T> &array, int index) {
59         return array.arr[index];
60     }
61 }
```

```

61
62 template <typename T>
63 void set(DynamicArray<T> &array, int index, T value) {
64     array.arr[index] = value;
65 }
66
67 template <typename T>
68 void print(const DynamicArray<T> &array, const string &separator) {
69     for (int i = 0; i < array.size; i++) {
70         cout << array.arr[i] << separator;
71     }
72     cout << endl;
73 }
74
75 int main() {
76     DynamicArray<int> arr;
77     int q;
78     cin >> q;
79     while (q--) {
80         string line;
81         cin >> line;
82         if (line == "insert") {
83             int index, N;
84             cin >> index >> N;
85             int *temp = new int[N];
86             for (int i = 0; i < N; i++) {
87                 cin >> temp[i];
88             }
89             insert(arr, index, N, temp);
90             delete[] temp;
91         } else if (line == "erase") {
92             int index, N;
93             cin >> index >> N;
94             erase(arr, index, N);
95         } else if (line == "size") {
96             cout << arr.size << endl;
97         } else if (line == "capacity") {
98             cout << arr.capacity << endl;
99         } else if (line == "get") {
100             int i;
101             cin >> i;
102             cout << get(arr, i) << endl;
103         } else if (line == "set") {
104             int i, value;
105             cin >> i >> value;
106             set(arr, i, value);
107         } else if (line == "print") {
108             print(arr, " ");
109         }
110     }
111     return 0;
112 }

```

algotester_lab_78_task_2_anhelina_bobrynok

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  template <typename T>
7  class DynamicArray {
8  private:
9      T *arr;
10     int size;
11     int capacity;
12
13 public:
14     // Конструктор
15     DynamicArray() {
16         size = 0;
17         capacity = 1;
18         arr = new T[capacity];
19     }
20     // Деструктор
21     ~DynamicArray() {
22         delete[] arr;
23     }
24     // Вставка елементів
25     void insert(int index, int amount, T *toInsert) {
26         while (size + amount >= capacity) {
27             capacity *= 2;
28         }
29         T *temp = new T[capacity];
30         for (int i = 0; i < index; i++) {
31             temp[i] = arr[i];
32         }

```

```

33         for (int i = 0; i < amount; i++) {
34             temp[index + i] = toInsert[i];
35         }
36         for (int i = index; i < size; i++) {
37             temp[i + amount] = arr[i];
38         }
39         size += amount;
40         delete[] arr;
41         arr = temp;
42     }
43     // Видалення елементів
44     void erase(int index, int amount) {
45         T *temp = new T[capacity];
46         int acc = 0;
47
48         for (int i = 0; i < size; i++) {
49             if (i < index || i >= index + amount) {
50                 temp[acc] = arr[i];
51                 acc++;
52             }
53         }
54
55         size -= amount;
56         delete[] arr;
57         arr = temp;
58     }
59     // Отримання елемента
60     T get(int index) const {

```



```

60     T get(int index) const {
61         return arr[index];
62     }
63     // Встановлення значення елемента
64     void set(int index, T value) {
65         arr[index] = value;
66     }
67     // Отримання розміру масиву
68     int getSize() const {
69         return size;
70     }
71     // Отримання ємності масиву
72     int getCapacity() const {
73         return capacity;
74     }
75     // Виведення масиву
76     void print(const string &separator) const {
77         for (int i = 0; i < size; i++) {
78             cout << arr[i] << separator;
79         }
80         cout << endl;
81     }
82 };
83
84 int main() {
85     DynamicArray<int> arr;
86     int q;
87     cin >> q;

```

```

87     cin >> q;
88     while (q-- > 0) {
89         string line;
90         cin >> line;
91         if (line == "insert") {
92             int index, N;
93             cin >> index >> N;
94             int *temp = new int[N];
95             for (int i = 0; i < N; i++) {
96                 cin >> temp[i];
97             }
98             arr.insert(index, N, temp);
99             delete[] temp;
100         } else if (line == "erase") {
101             int index, N;
102             cin >> index >> N;
103             arr.erase(index, N);
104         } else if (line == "size") {
105             cout << arr.getSize() << endl;
106         } else if (line == "capacity") {
107             cout << arr.getCapacity() << endl;
108         } else if (line == "get") {
109             int i;
110             cin >> i;
111             cout << arr.get(i) << endl;
112         } else if (line == "set") {
113             int i, value;
114             cin >> i >> value;
115             arr.set(i, value);
116         } else if (line == "print") {

```

Завдання №4

practice_work_task_1_anhelina_bobrynok.cpp

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  template <typename T>
7  class DynamicArray {
8  private:
9      T *arr;
10     int size;
11     int capacity;
12
13 public:
14     // Конструктор
15     DynamicArray() {
16         size = 0;
17         capacity = 1;
18         arr = new T[capacity];
19     }
20     // Деструктор
21     ~DynamicArray() {
22         delete[] arr;
23     }
24     // Вставка елементів
25     void insert(int index, int amount, T *toInsert) {
26         while (size + amount >= capacity) {
27             capacity *= 2;
28         }
29         T *temp = new T[capacity];
30         for (int i = 0; i < index; i++) {
31             temp[i] = arr[i];
32
33             for (int i = 0; i < amount; i++) {
34                 temp[index + i] = toInsert[i];
35             }
36             for (int i = index; i < size; i++) {
37                 temp[i + amount] = arr[i];
38             }
39             size += amount;
40             delete[] arr;
41             arr = temp;
42         }
43         // Видалення елементів
44         void erase(int index, int amount) {
45             T *temp = new T[capacity];
46             int acc = 0;
47
48             for (int i = 0; i < size; i++) {
49                 if (i < index || i >= index + amount) {
50                     temp[acc] = arr[i];
51                     acc++;
52                 }
53             }
54
55             size -= amount;
56             delete[] arr;
57             arr = temp;
58         }

```

```
59     // Отримання елемента
60     T get(int index) const {
61         return arr[index];
62     }
63     // Встановлення значення елемента
64     void set(int index, T value) {
65         arr[index] = value;
66     }
67     // Отримання розміру масиву
68     int getSize() const {
69         return size;
70     }
71     // Отримання ємності масиву
72     int getCapacity() const {
73         return capacity;
74     }
75     // Виведення масиву
76     void print(const string &separator) const {
77         for (int i = 0; i < size; i++) {
78             cout << arr[i] << separator;
79         }
80         cout << endl;
81     }
82 };
83
84 int main() {
85     DynamicArray<int> arr;
86     int q;
87     cin >> q;
```

```

87     cin >> q;
88     while (q-->0) {
89         string line;
90         cin >> line;
91         if (line == "insert") {
92             int index, N;
93             cin >> index >> N;
94             int *temp = new int[N];
95             for (int i = 0; i < N; i++) {
96                 cin >> temp[i];
97             }
98             arr.insert(index, N, temp);
99             delete[] temp;
100         } else if (line == "erase") {
101             int index, N;
102             cin >> index >> N;
103             arr.erase(index, N);
104         } else if (line == "size") {
105             cout << arr.getSize() << endl;
106         } else if (line == "capacity") {
107             cout << arr.getCapacity() << endl;
108         } else if (line == "get") {
109             int i;
110             cin >> i;
111             cout << arr.get(i) << endl;
112         } else if (line == "set") {
113             int i, value;
114             cin >> i >> value;
115             arr.set(i, value);
116         } else if (line == "print") {

```

practice_work_task_2_anhelina_bobrynok.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  class TreeNode {
5  public:
6      int data;
7      TreeNode* left;
8      TreeNode* right;
9
10     TreeNode(int value) {
11         data = value;
12         left = nullptr;
13         right = nullptr;
14     }
15 };
16
17 class BinaryTree {
18     TreeNode* root;
19
20 public:
21     BinaryTree() {
22         root = nullptr;
23     }
24
25     void insert(int value) {
26         root = insert(root, value);
27     }
28
29     TreeNode* insert(TreeNode* node, int value) {
30         if (node == nullptr) {
31             return new TreeNode(value);
32         }
```

```
33
34     if (value < node->data) {
35         node->left = insert(node->left, value);
36     } else if (value > node->data) {
37         node->right = insert(node->right, value);
38     }
39
40     return node;
41 }
42
43 void inOrder() {
44     inOrder(root);
45     cout << endl;
46 }
47
48 void inOrder(TreeNode* node) {
49     if (node != nullptr) {
50         inOrder(node->left);
51         cout << node->data << " ";
52         inOrder(node->right);
53     }
54 }
55
56 TreeNode* mirrorFlip(TreeNode* node) {
57     if (node == nullptr) {
58         return nullptr;
59     }
60
```

```

61     TreeNode* flipped = new TreeNode(node->data);
62     flipped->left = mirrorFlip(node->right);
63     flipped->right = mirrorFlip(node->left);
64
65     return flipped;
66 }
67
68 void sumSubtrees(TreeNode* node) {
69     if (node == nullptr) return;
70
71     sumSubtrees(node->left);
72     sumSubtrees(node->right);
73
74     if (node->left != nullptr) {
75         node->data += node->left->data;
76     }
77     if (node->right != nullptr) {
78         node->data += node->right->data;
79     }
80 }
81
82 void sumSubtrees() {
83     sumSubtrees(root);
84 }
85
86 BinaryTree mirror() {
87     BinaryTree mirroredTree;
88     mirroredTree.root = mirrorFlip(root);
89     return mirroredTree;
90 }

```



```

92     void displayTree(TreeNode* node) {
93         if (node == nullptr) return;
94
95         cout << node->data << " ";
96         displayTree(node->left);
97         displayTree(node->right);
98     }
99
100    void displayTree() {
101        displayTree(root);
102        cout << endl;
103    }
104 };
105
106 int main() {
107     BinaryTree tree;
108     tree.insert(10);
109     tree.insert(5);
110     tree.insert(15);
111     tree.insert(3);
112     tree.insert(7);
113
114     cout << "Original tree (In-order traversal): ";
115     tree.inOrder();
116
117     cout << "\nMirrored tree: ";
118     BinaryTree mirroredTree = tree.mirror();
119     mirroredTree.displayTree();
120
121     cout << "\nTree after adding subtree sums: ";
122
123     tree.sumSubtrees();
124     tree.displayTree();
125
126     return 0;
127 }

```

Завдання №5

self_practice_work_anhelina_bobrynok.cpp

```

// Функція для перевірки, чи є слово паліндромом
bool isPalindrome(const string& word) {
    string lowerWord = word;
    transform(lowerWord.begin(), lowerWord.end(), lowerWord.begin(), ::tolower); // Приведення до нижнього регістру
    string reversedWord = lowerWord;
    reverse(reversedWord.begin(), reversedWord.end()); // Створення оберненої копії
    return lowerWord == reversedWord; // Перевірка на рівність
}

int main() {
    string line;
    int palindromeCount = 0;

    // Зчитуємо рядки до кінця вхідних даних
    while (getline(cin, line)) {
        stringstream ss(line);
        string word;

        // Розбиваємо рядок на слова
        while (ss >> word) {
            if (isPalindrome(word)) {
                palindromeCount++;
            }
        }
    }

    // Виводимо результат
    cout << palindromeCount << endl;
    return 0;
}

```

5) Результати виконання завдань, тестування та фактично затрачений час:

Завдання №1

```

10 20 30 40 50
10 40 50
10 25 40 50
Список порожній.
10 25 40 50
Список порожній.

```

Завдання №2

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.025	1.793	Перегляд

Завдання №3

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.006	1.273	Перегляд

Завдання №4

```
Original lists:
6 -> 4 -> 1 -> 2 -> 3 -> 11 -> null
9 -> 0 -> 7 -> 8 -> 4 -> 1 -> null
Reversed lists:
11 -> 3 -> 2 -> 1 -> 4 -> 6 -> null
1 -> 4 -> 8 -> 7 -> 0 -> 9 -> null
Comparing lists: Lists are different.
Adding lists:
Number of sum is: 1549082
1 -> 5 -> 4 -> 9 -> 0 -> 8 -> 2 -> null
```

```
Original tree (In-order traversal): 3 5 7 10 15
Mirrored tree: 10 15 5 7 3
Tree after adding subtree sums: 40 15 3 7 15
```

Завдання №5

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.003	1.203	Перегляд

Висновки: Завдяки цій лабораторній роботі я мала змогу навчитись працювати з динамічними структурами, реалізувати зв'язний список та навчитись працювати з бінарними деревами.