Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHub yang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. Asisten Praktikum : 4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folder dalam repository akan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwal praktikum adalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktu ini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem-solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

8. **Struktur Laporan Praktikum**

1. Cover :

LAPORAN PRAKTIKUM
Modul 7
“Stack”

Disusun Oleh:
Reza Afiansyah Wibowo -2311104062
SE0702

Dosen :
WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

2. Tujuan

a.Mampu memahami konsep stack pada struktur data dan algoritma
b.Mampu mengimplementasikan operasi-operasi pada stack
c.Mampu memecahkan permasalahan dengan solusi stack

3. Landasan Teori

Sebuah stack atau tumpukan merupakan struktur data yang berfungsi untuk menyimpan dan mengelola kumpulan data dengan prinsip Last-In, First-Out

(LIFO). Analogi yang sering digunakan adalah tumpukan piring di kafetaria, di mana piring terakhir yang ditambahkan akan menjadi yang pertama diambil.

Dalam implementasinya, stack dapat direpresentasikan sebagai struktur data terurut yang memiliki dua operasi utama: push dan pop. Operasi push digunakan untuk menambahkan elemen baru ke dalam stack, sementara operasi pop digunakan untuk menghapus elemen teratas dari stack.

Prinsip LIFO yang menjadi dasar stack membuatnya sangat bermanfaat dalam berbagai aplikasi, termasuk manajemen memori komputer, evaluasi ekspresi aritmatika, dan manajemen panggilan fungsi dalam pemrograman. Sebagai contoh, dalam manajemen memori, stack digunakan untuk menyimpan alamat-alamat memori yang dialokasikan untuk variabel dan fungsi.

Dengan prinsip LIFO ini, stack memungkinkan akses data dengan efisiensi, di mana elemen yang terakhir dimasukkan akan menjadi yang pertama diambil. Hal ini menjadikannya salah satu struktur data yang sangat penting dalam pengembangan perangkat lunak dan pemrograman komputer..

4. Guided

Guided 1

```
#include <iostream>
#define MAX 100

using namespace std;

class Stack {
private:
    int top;
    int arr[MAX];

public:
    Stack() { top = -1; }

    bool isFull() { return top == MAX - 1; }
    bool isEmpty() { return top == -1; }

    void push(int x) {
        if (isFull()) {
            cout << "Stack Overflow\n";
            return;
        }
        arr[++top] = x;
    }

    void pop() {
```

```
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        top--;
    }

    int peek() {
        if (!isEmpty()) {
            return arr[top];
        }
        cout << "Stack is empty\n";
        return -1; // Return a sentinel value
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << "\n";
    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    s.pop();
    cout << "After popping, stack elements: ";
    s.display();
```

```
    return 0;
}
```

Guided 2

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

class Node {
public:
    int data;
    Node* next;

    Node(int value) {
        data = value;
        next = nullptr;
    }
};

class Stack {
private:
    Node* top;

public:
    Stack() { top = nullptr; }

    bool isEmpty() { return top == nullptr; }

    void push(int x) {
        Node* newNode = new Node(x);
        newNode->next = top;
        top = newNode;
    }

    void pop() {
        if (isEmpty()) {
            cout << "Stack Underflow\n";
            return;
        }
        Node* temp = top;
        top = top->next;
        delete temp;
```

```
    }

    int peek() {
        if (!isEmpty()) {
            return top->data;
        }
        cout << "Stack is empty\n";
        return -1; // Return a sentinel value
    }

    void display() {
        if (isEmpty()) {
            cout << "Stack is empty\n";
            return;
        }
        Node* current = top;
        while (current) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << "\n";
    }
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    s.push(30);

    cout << "Stack elements: ";
    s.display();

    cout << "Top element: " << s.peek() << "\n";

    s.pop();
    cout << "After popping, stack elements: ";
    s.display();

    return 0;
}
```

5. Unguided

Nomor 1

```
#include <iostream>
#include <string>
#include <stack>
#include <algorithm>

using namespace std;

// Function to check if a sentence is a palindrome
bool isPalindrome(string sentence) {
    // Remove spaces and convert to lowercase
    string cleanSentence;
    for (char c : sentence) {
        if (!isspace(c)) {
            cleanSentence += tolower(c);
        }
    }

    // Create a stack to store characters
    stack<char> charStack;

    // Push first half of characters to stack
    int mid = cleanSentence.length() / 2;
    for (int i = 0; i < mid; i++) {
        charStack.push(cleanSentence[i]);
    }

    // Skip middle character for odd-length strings
    int startIndex = (cleanSentence.length() % 2 == 0) ? mid : mid + 1;

    // Compare second half with stack
    for (int i = startIndex; i < cleanSentence.length(); i++) {
        // If stack top doesn't match current character, not a palindrome
        if (charStack.top() != cleanSentence[i]) {
            return false;
        }
        charStack.pop();
    }

    return true;
}
```

```
int main() {
    string sentence;

    cout << "Masukkan kalimat: ";
    getline(cin, sentence);

    if (isPalindrome(sentence)) {
        cout << "Kalimat adalah palindrom." << endl;
    } else {
        cout << "Kalimat bukan palindrom." << endl;
    }

    return 0;
}
```

Nomor 2

```
#include <iostream>
#include <stack>
#include <sstream>
#include <string>

using namespace std;

// Function to reverse a sentence using stack
string reverseSentence(const string& sentence) {
    // Create a stack to store words
    stack<string> wordStack;

    // Use stringstream to split sentence into words
    istringstream iss(sentence);
    string word;

    // Push each word onto the stack
    while (iss >> word) {
        wordStack.push(word);
    }

    // Rebuild the sentence by popping words from the stack
    string reversedSentence;
    while (!wordStack.empty()) {
        // Add word to reversed sentence
        reversedSentence += wordStack.top() + " ";
```

```
        // Remove the top word from the stack
        wordStack.pop();
    }

    // Remove trailing space and return
    if (!reversedSentence.empty()) {
        reversedSentence.pop_back();
    }

    return reversedSentence;
}

int main() {
    string sentence;

    // Prompt for input with at least 3 words
    while (true) {
        cout << "Masukkan kalimat (minimal 3 kata): ";
        getline(cin, sentence);

        // Count words
        istringstream wordCounter(sentence);
        int wordCount = 0;
        string temp;
        while (wordCounter >> temp) {
            wordCount++;
        }

        // Validate input
        if (wordCount >= 3) {
            break;
        }

        cout << "Kalimat harus memiliki minimal 3 kata. Silakan coba lagi." << endl;
    }

    // Reverse the sentence
    string reversedSentence = reverseSentence(sentence);

    // Output results
    cout << "Kalimat asli: " << sentence << endl;
    cout << "Kalimat dibalik: " << reversedSentence << endl;
```

```
    return 0;
}
```

6. Kesimpulan

Stack adalah struktur data yang mengikuti prinsip Last-In, First-Out (LIFO), di mana elemen terakhir yang dimasukkan adalah yang pertama kali dikeluarkan. Konsep ini mirip dengan tumpukan piring di kafetaria, di mana piring terakhir yang ditaruh adalah piring pertama yang diambil.