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:
KAFKA PUTRA RIYADI-2311104041

Kelas:
SE-07-02

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng,

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

Stack adalah salah satu struktur data linear yang berfungsi sebagai tempat penyimpanan elemen dengan prinsip LIFO (Last In, First Out), yaitu elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Stack banyak digunakan dalam berbagai aplikasi, seperti pengelolaan memori, evaluasi ekspresi, dan algoritma rekursi.

Pada Stack terdapat beberapa operasi utama diantaranya yaitu:

- Push: Menambahkan elemen ke bagian atas stack.
- Pop: Menghapus elemen dari bagian atas stack.
- Peek (atau Top): Melihat elemen di bagian atas stack tanpa menghapusnya.
- IsEmpty: Mengecek apakah stack kosong.
- IsFull: Mengecek apakah stack penuh (dalam implementasi berbasis array).

4. Guided

Code Program Guided 1:

```cpp
#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;
    }

    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;
}
```

Kode di atas adalah implementasi struktur data Stack yang menunjukkan bagaimana operasi dasar stack bekerja yang diman pada codingan tersebut dapat melakukan beberapa hal yaitu :

a. Membuat stack menggunakan array.
b. Melakukan operasi dasar stack: push, pop, peek, dan display.
c. Memanfaatkan prinsip LIFO (Last In, First Out) dalam pengelolaan elemen.

Outputannya:

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS D:\STRUKTUR DATA P7\output>
```

Code Program Guided 2:

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

    return 0;
}
```

Tujuan dari codingan di atas adalah mempelajari dan memahami implementasi stack menggunakan linked list serta mendemonstrasikan operasi-operasi dasar stack yang sesuai dengan prinsip LIFO dalam struktur data dinamis. Yang dimana node dalam linked list dibuat secara dinamis menggunakan new, dan bagaimana node tersebut dihubungkan menggunakan pointer (next).

Outputannya:

```
Stack elements: 30 20 10
Top element: 30
After popping, stack elements: 10
PS D:\STRUKTUR DATA P7\output>
```

5. Unguided

1.

```cpp
#include <iostream>
#include <stack>
#include <cctype>
using namespace std;
bool isPalindrome(string sentence) {
    stack<char> charStack;
    string normalized = "";

    for (char c : sentence) {
        if (isalnum(c)) {
            normalized += tolower(c);
        }
    }

    for (char c : normalized) {
        charStack.push(c);
    }

    string reversed = "";
    while (!charStack.empty()) {
        reversed += charStack.top();
        charStack.pop();
    }

    return normalized == reversed;
}

int main() {
    string sentence;

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

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

    return 0;
}

```

Outputannya:

```
Masukkan kalimat: ini
Kalimat 'ini' adalah palindrom.
PS D:\STRUKTUR DATA P7\output>
```

```
Masukkan kalimat: Telkom
Kalimat 'Telkom' bukan palindrom.
PS D:\STRUKTUR DATA P7\output>
```

2.

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

int main() {

    stack<char> charStack;
    string sentence, reversedSentence = "";

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

    for (char c : sentence) {
        charStack.push(c);
    }

    while (!charStack.empty()) {
        reversedSentence += charStack.top();
        charStack.pop();
    }

    cout << "Hasil: " << reversedSentence << endl;

    return 0;
}

```

Outputannnya:

```
Masukkan kalimat: Telkom Purwokerto
Hasil: otrekowruP mokleT
PS D:\STRUKTUR DATA P7\output>
```

6. Kesimpulan

Stack adalah struktur data linear berbasis **LIFO (Last In, First Out)**, di mana elemen terakhir yang masuk akan keluar pertama. Operasi utamanya meliputi:

- Push: Menambahkan elemen.
- Pop: Menghapus elemen.
- Peek: Melihat elemen atas tanpa menghapusnya.
- IsEmpty: Mengecek apakah kosong.
- IsFull: Mengecek apakah penuh.

Stack banyak digunakan dalam pengelolaan memori, evaluasi ekspresi, dan algoritma rekursi.