LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 10
PENGENALAN CODE BLOCKS

Disusun Oleh :
NAMA : Jauza Rasyiq Hernanta
NIM : **103112430033**

Dosen
WAHYU ANDI SAPUTRA

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

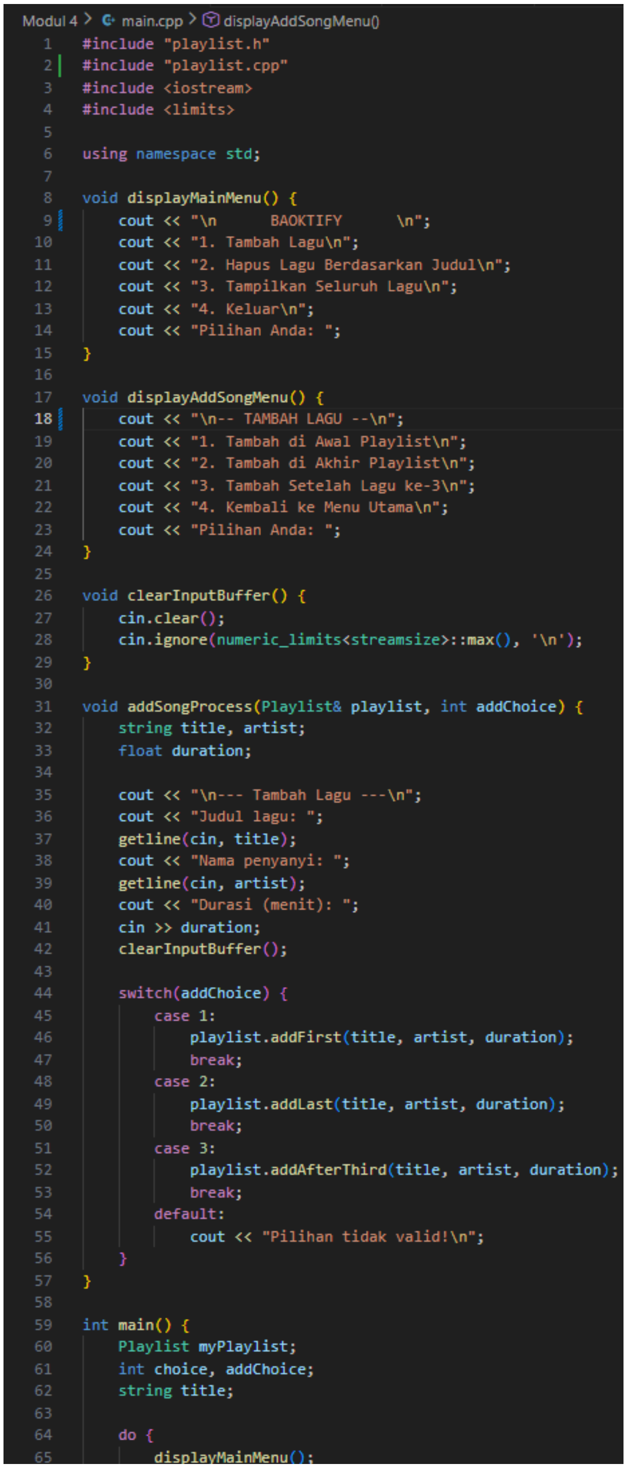
A. Dasar Teori

Stack merupakan salah satu bentuk struktur data dimana prinsip operasi yang digunakan seperti tumpukan. Seperti halnya tumpukan, elemen yang bisa diambil terlebih dahulu adalah elemen yang paling atas, atau elemen yang pertama kali masuk, prinsip ini biasa disebut LIFO (Last In First Out).

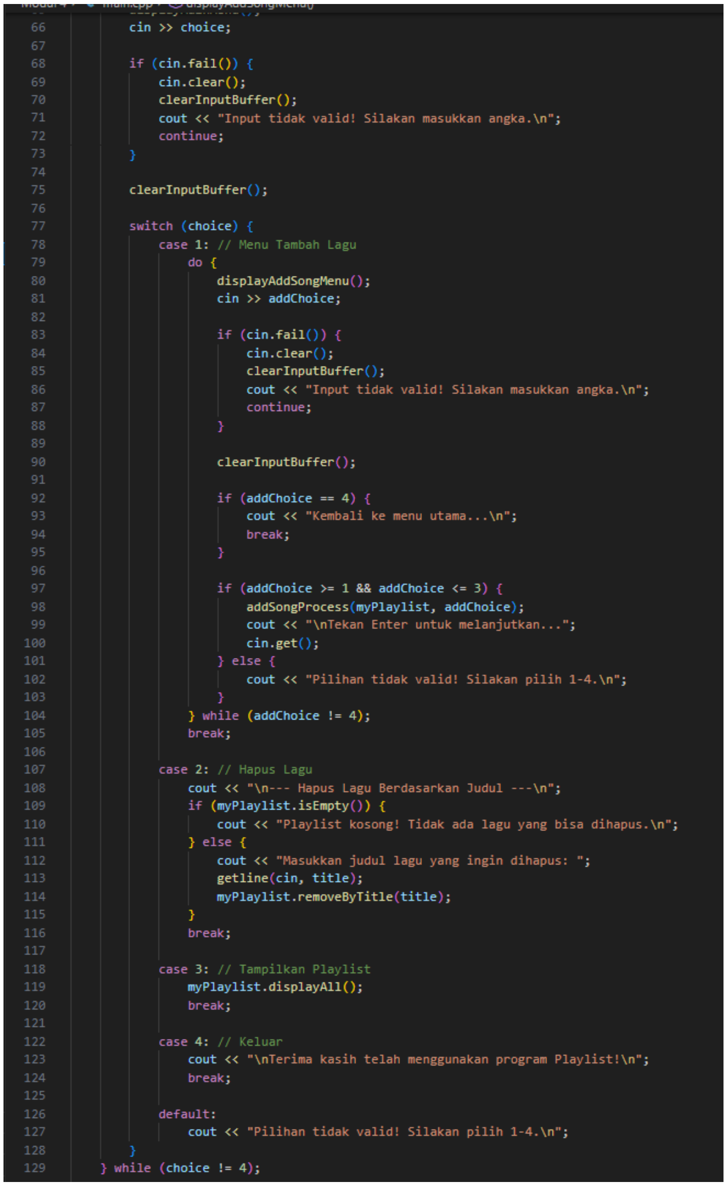
B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

Main.cpp

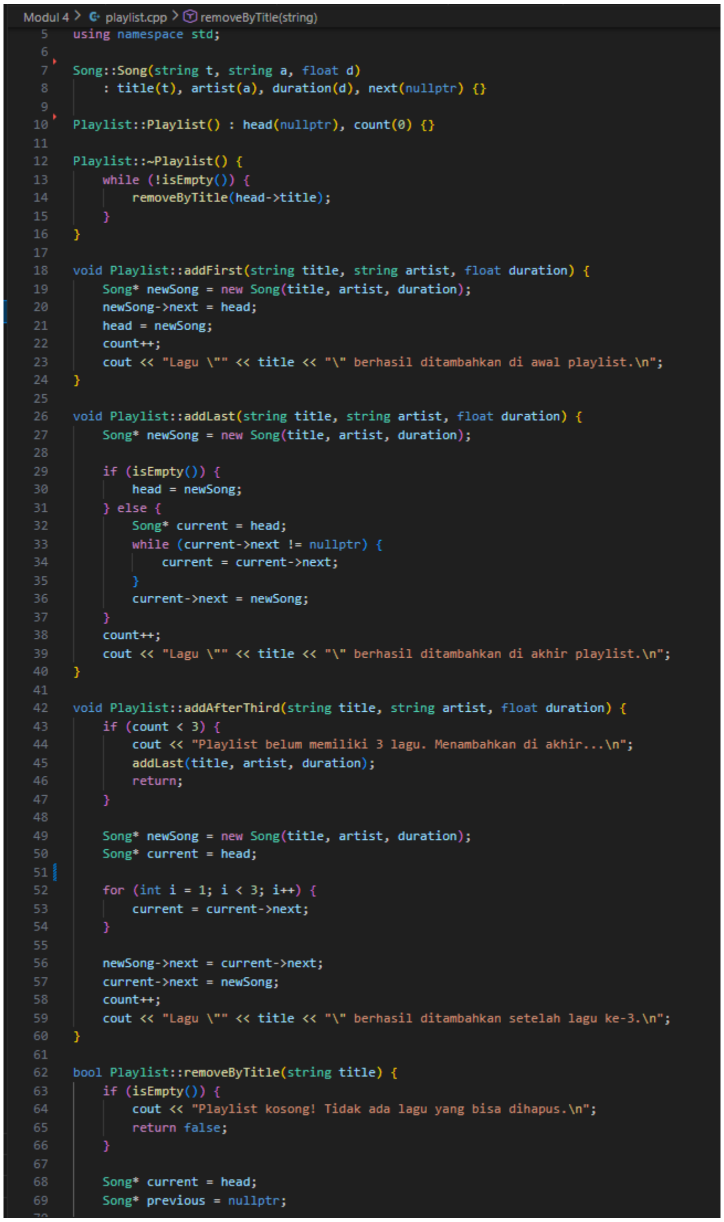


```cpp
#include "playlist.h"
#include "playlist.cpp"
#include <iostream>
#include <limits>

using namespace std;

void displayMainMenu() {
    cout << "\n      BAOKTIFY      \n";
    cout << "1. Tambah Lagu\n";
    cout << "2. Hapus Lagu Berdasarkan Judul\n";
    cout << "3. Tampilkan Seluruh Lagu\n";
    cout << "4. Keluar\n";
    cout << "Pilihan Anda: ";
}

void displayAddSongMenu() {
    cout << "\n-- TAMBAH LAGU --\n";
    cout << "1. Tambah di Awal Playlist\n";
    cout << "2. Tambah di Akhir Playlist\n";
    cout << "3. Tambah Setelah Lagu ke-3\n";
    cout << "4. Kembali ke Menu Utama\n";
    cout << "Pilihan Anda: ";
}

void clearInputBuffer() {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

void addSongProcess(Playlist& playlist, int addChoice) {
    string title, artist;
    float duration;

    cout << "\n--- Tambah Lagu ---\n";
    cout << "Judul lagu: ";
    getline(cin, title);
    cout << "Nama penyanyi: ";
    getline(cin, artist);
    cout << "Durasi (menit): ";
    cin >> duration;
    clearInputBuffer();

    switch(addChoice) {
        case 1:
            playlist.addFirst(title, artist, duration);
            break;
        case 2:
            playlist.addLast(title, artist, duration);
            break;
        case 3:
            playlist.addAfterThird(title, artist, duration);
            break;
        default:
            cout << "Pilihan tidak valid!\n";
    }
}

int main() {
    Playlist myPlaylist;
    int choice, addChoice;
    string title;

    do {
        displayMainMenu();
```

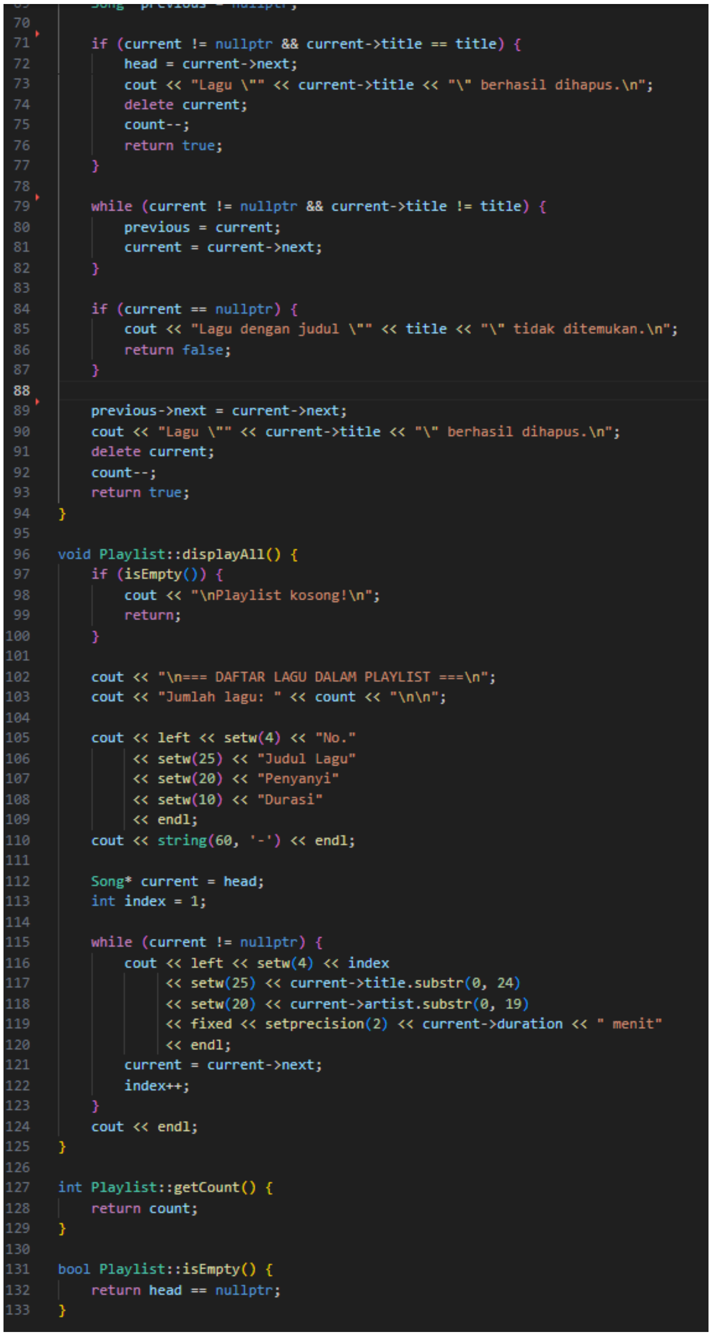


```cpp
        cin >> choice;

        if (cin.fail()) {
            cin.clear();
            clearInputBuffer();
            cout << "Input tidak valid! Silakan masukkan angka.\n";
            continue;
        }

        clearInputBuffer();

        switch (choice) {
            case 1: // Menu Tambah Lagu
                do {
                    displayAddSongMenu();
                    cin >> addChoice;

                    if (cin.fail()) {
                        cin.clear();
                        clearInputBuffer();
                        cout << "Input tidak valid! Silakan masukkan angka.\n";
                        continue;
                    }

                    clearInputBuffer();

                    if (addChoice == 4) {
                        cout << "Kembali ke menu utama...\n";
                        break;
                    }

                    if (addChoice >= 1 && addChoice <= 3) {
                        addSongProcess(myPlaylist, addChoice);
                        cout << "\nTekan Enter untuk melanjutkan...";
                        cin.get();
                    } else {
                        cout << "Pilihan tidak valid! Silakan pilih 1-4.\n";
                    }
                } while (addChoice != 4);
                break;

            case 2: // Hapus Lagu
                cout << "\n--- Hapus Lagu Berdasarkan Judul ---\n";
                if (myPlaylist.isEmpty()) {
                    cout << "Playlist kosong! Tidak ada lagu yang bisa dihapus.\n";
                } else {
                    cout << "Masukkan judul lagu yang ingin dihapus: ";
                    getline(cin, title);
                    myPlaylist.removeByTitle(title);
                }
                break;

            case 3: // Tampilkan Playlist
                myPlaylist.displayAll();
                break;

            case 4: // Keluar
                cout << "\nTerima kasih telah menggunakan program Playlist!\n";
                break;

            default:
                cout << "Pilihan tidak valid! Silakan pilih 1-4.\n";
        }
    } while (choice != 4);
```

Playlist,cpp

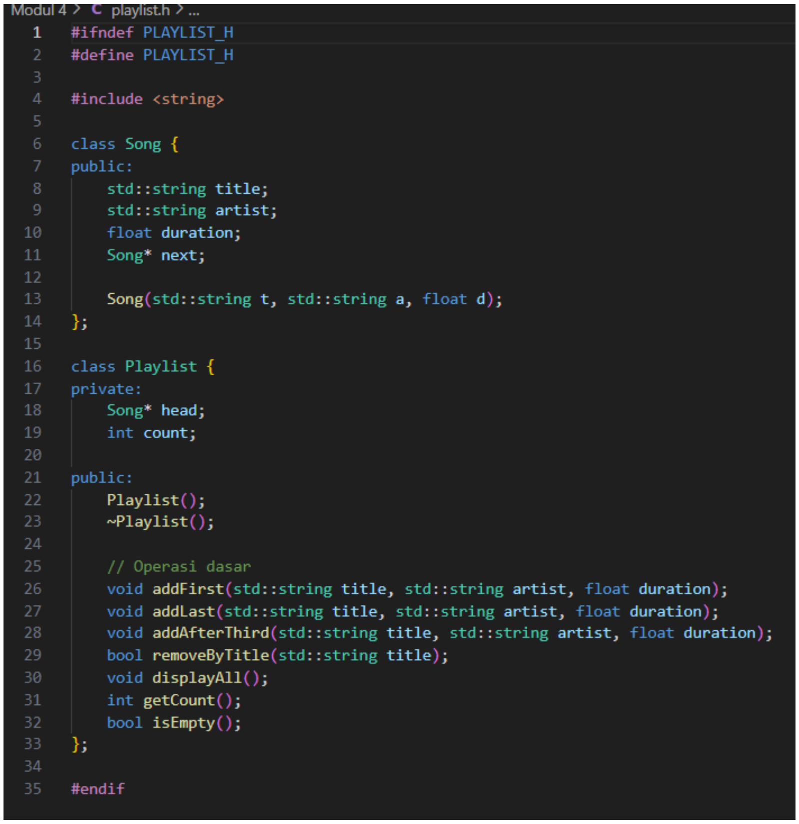


```cpp
using namespace std;

Song::Song(string t, string a, float d)
    : title(t), artist(a), duration(d), next(nullptr) {}

Playlist::Playlist() : head(nullptr), count(0) {}

Playlist::~Playlist() {
    while (!isEmpty()) {
        removeByTitle(head->title);
    }
}

void Playlist::addFirst(string title, string artist, float duration) {
    Song* newSong = new Song(title, artist, duration);
    newSong->next = head;
    head = newSong;
    count++;
    cout << "Lagu \"" << title << "\" berhasil ditambahkan di awal playlist.\n";
}

void Playlist::addLast(string title, string artist, float duration) {
    Song* newSong = new Song(title, artist, duration);

    if (isEmpty()) {
        head = newSong;
    } else {
        Song* current = head;
        while (current->next != nullptr) {
            current = current->next;
        }
        current->next = newSong;
    }
    count++;
    cout << "Lagu \"" << title << "\" berhasil ditambahkan di akhir playlist.\n";
}

void Playlist::addAfterThird(string title, string artist, float duration) {
    if (count < 3) {
        cout << "Playlist belum memiliki 3 lagu. Menambahkan di akhir...\n";
        addLast(title, artist, duration);
        return;
    }

    Song* newSong = new Song(title, artist, duration);
    Song* current = head;

    for (int i = 1; i < 3; i++) {
        current = current->next;
    }

    newSong->next = current->next;
    current->next = newSong;
    count++;
    cout << "Lagu \"" << title << "\" berhasil ditambahkan setelah lagu ke-3.\n";
}

bool Playlist::removeByTitle(string title) {
    if (isEmpty()) {
        cout << "Playlist kosong! Tidak ada lagu yang bisa dihapus.\n";
        return false;
    }

    Song* current = head;
    Song* previous = nullptr;
```



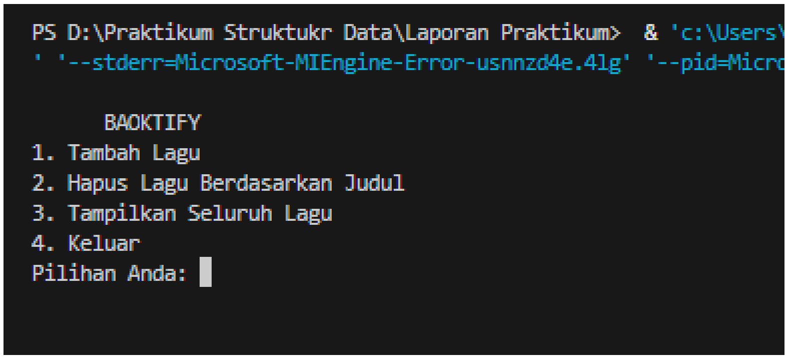
```cpp
    Song* previous = nullptr;

    if (current != nullptr && current->title == title) {
        head = current->next;
        cout << "Lagu \"" << current->title << "\" berhasil dihapus.\n";
        delete current;
        count--;
        return true;
    }

    while (current != nullptr && current->title != title) {
        previous = current;
        current = current->next;
    }

    if (current == nullptr) {
        cout << "Lagu dengan judul \"" << title << "\" tidak ditemukan.\n";
        return false;
    }

    previous->next = current->next;
    cout << "Lagu \"" << current->title << "\" berhasil dihapus.\n";
    delete current;
    count--;
    return true;
}

void Playlist::displayAll() {
    if (isEmpty()) {
        cout << "\nPlaylist kosong!\n";
        return;
    }

    cout << "\n=== DAFTAR LAGU DALAM PLAYLIST ===\n";
    cout << "Jumlah lagu: " << count << "\n\n";

    cout << left << setw(4) << "No."
         << setw(25) << "Judul Lagu"
         << setw(20) << "Penyanyi"
         << setw(10) << "Durasi"
         << endl;
    cout << string(60, '-') << endl;

    Song* current = head;
    int index = 1;

    while (current != nullptr) {
        cout << left << setw(4) << index
             << setw(25) << current->title.substr(0, 24)
             << setw(20) << current->artist.substr(0, 19)
             << fixed << setprecision(2) << current->duration << " menit"
             << endl;
        current = current->next;
        index++;
    }
    cout << endl;
}

int Playlist::getCount() {
    return count;
}

bool Playlist::isEmpty() {
    return head == nullptr;
}
```

Playlist.h

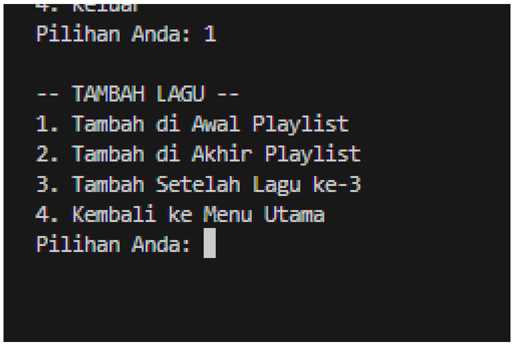


```cpp
#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <string>

class Song {
public:
    std::string title;
    std::string artist;
    float duration;
    Song* next;

    Song(std::string t, std::string a, float d);
};

class Playlist {
private:
    Song* head;
    int count;

public:
    Playlist();
    ~Playlist();

    // Operasi dasar
    void addFirst(std::string title, std::string artist, float duration);
    void addLast(std::string title, std::string artist, float duration);
    void addAfterThird(std::string title, std::string artist, float duration);
    bool removeByTitle(std::string title);
    void displayAll();
    int getCount();
    bool isEmpty();
};

#endif
```

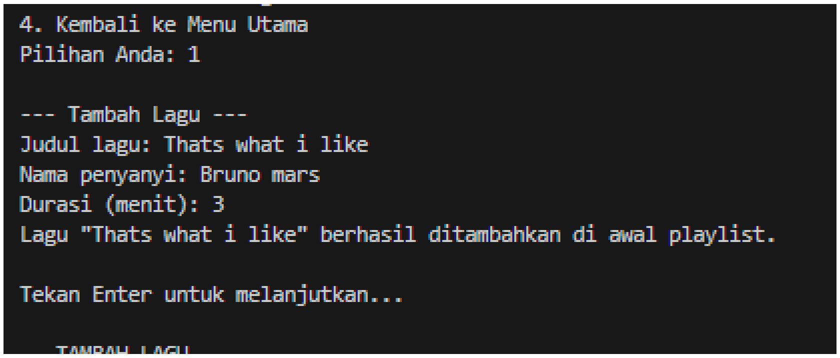
Screenshots Output



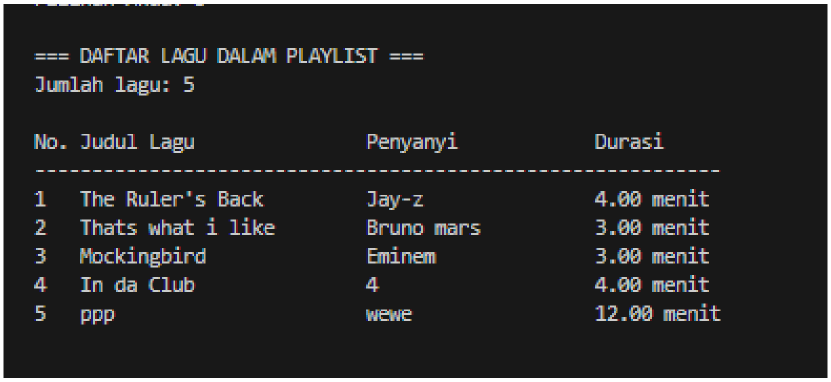
```
PS D:\Praktikum Struktukr Data\Laporan Praktikum>  & 'c:\Users\
' '--stderr=Microsoft-MIEngine-Error-usnnzd4e.4lg' '--pid=Micro

      BAOKTIFY
1. Tambah Lagu
2. Hapus Lagu Berdasarkan Judul
3. Tampilkan Seluruh Lagu
4. Keluar
Pilihan Anda: 
```



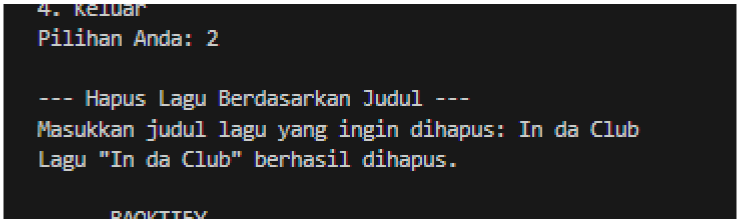
```
4. Keluar
Pilihan Anda: 1

-- TAMBAH LAGU --
1. Tambah di Awal Playlist
2. Tambah di Akhir Playlist
3. Tambah Setelah Lagu ke-3
4. Kembali ke Menu Utama
Pilihan Anda: 
```



```
4. Kembali ke Menu Utama
Pilihan Anda: 1

--- Tambah Lagu ---
Judul lagu: Thats what i like
Nama penyanyi: Bruno mars
Durasi (menit): 3
Lagu "Thats what i like" berhasil ditambahkan di awal playlist.

Tekan Enter untuk melanjutkan...

   TAMBAH LAGU
```



```
 Pilihan Anda: 1

 === DAFTAR LAGU DALAM PLAYLIST ===
 Jumlah lagu: 5

 No. Judul Lagu              Penyanyi           Durasi
 -----------------------------------------------------------
 1   The Ruler's Back        Jay-z              4.00 menit
 2   Thats what i like       Bruno mars         3.00 menit
 3   Mockingbird             Eminem             3.00 menit
 4   In da Club              4                  4.00 menit
 5   ppp                     wewe               12.00 menit
```



```
 4. Keluar
 Pilihan Anda: 2

 --- Hapus Lagu Berdasarkan Judul ---
 Masukkan judul lagu yang ingin dihapus: In da Club
 Lagu "In da Club" berhasil dihapus.

         BACKTIFY
```