Nama : Ahmad Fadillah Noor
Nim : 2209076023

Laporan Penjelasan Soal UTS Struktur Data

1. Soal Array dan Pointer

Penjelasan program yang digunakan

1. Struct Mahasiswa: Terdiri dari NIM, nama, dan IPK.
2. Array of Pointers dataMahasiswa[MAX_MAHASISWA]: Menyimpan data mahasiswa menggunakan array pointer.
3. Fungsi tambahMahasiswa: Menambah mahasiswa baru ke array.
4. Fungsi hapusMahasiswa: Menghapus data mahasiswa berdasarkan NIM yang diinputkan.
5. Fungsi tampilkanMahasiswa: Menampilkan seluruh data mahasiswa yang ada.
6. Fungsi urutkanMahasiswaByIPK: Mengurutkan mahasiswa berdasarkan IPK menggunakan algoritma Bubble Sort.
7. Main Menu: Menampilkan menu pilihan dan menerima input pengguna.

Cara Menggunakan Program

1. Pilih menu Tambah Mahasiswa untuk memasukkan data mahasiswa.
2. Gunakan menu Hapus Mahasiswa untuk menghapus data mahasiswa.
3. Pilih Tampilkan Mahasiswa untuk melihat daftar mahasiswa.
4. Pilih Urutkan Mahasiswa berdasarkan IPK untuk mengurutkan data berdasarkan IPK tertinggi.

Codingan

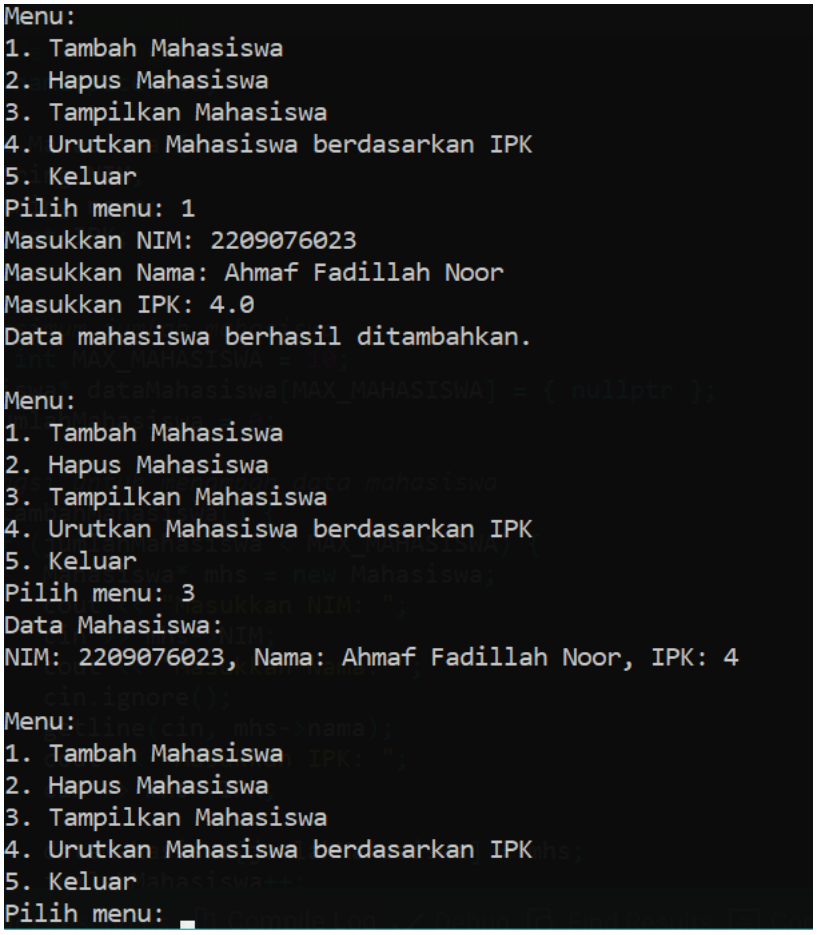
```
#include <iostream>
#include <string>
using namespace std;

struct Mahasiswa {
    string NIM;
    string nama;
    float IPK;
};

// Maksimum jumlah mahasiswa
const int MAX_MAHASISWA = 10;
Mahasiswa* dataMahasiswa[MAX_MAHASISWA] = { nullptr };
```

```
int jumlahMahasiswa = 0;

// Fungsi untuk menambah data mahasiswa
void tambahMahasiswa() {
    if (jumlahMahasiswa < MAX_MAHASISWA) {
        Mahasiswa* mhs = new Mahasiswa;
        cout << "Masukkan NIM: ";
        cin >> mhs->NIM;
        cout << "Masukkan Nama: ";
        cin.ignore();
        getline(cin, mhs->nama);
        cout << "Masukkan IPK: ";
        cin >> mhs->IPK;

        dataMahasiswa[jumlahMahasiswa] = mhs;
        jumlahMahasiswa++;
        cout << "Data mahasiswa berhasil ditambahkan.\n";
    } else {
        cout << "Data mahasiswa sudah penuh.\n";
    }
}

// Fungsi untuk menghapus data mahasiswa berdasarkan NIM
void hapusMahasiswa() {
    string nim;
    cout << "Masukkan NIM mahasiswa yang ingin dihapus: ";
    cin >> nim;
    bool ditemukan = false;

    for (int i = 0; i < jumlahMahasiswa; i++) {
        if (dataMahasiswa[i] != nullptr && dataMahasiswa[i]->NIM ==
nim) {
            delete dataMahasiswa[i];
            dataMahasiswa[i] = nullptr;
            for (int j = i; j < jumlahMahasiswa - 1; j++) {
                dataMahasiswa[j] = dataMahasiswa[j + 1];
            }
            dataMahasiswa[jumlahMahasiswa - 1] = nullptr;
            jumlahMahasiswa--;
            ditemukan = true;
            cout << "Data mahasiswa berhasil dihapus.\n";
            break;
        }
    }
    if (!ditemukan) {
        cout << "Mahasiswa dengan NIM " << nim << " tidak
ditemukan.\n";
    }
}

// Fungsi untuk menampilkan data mahasiswa
void tampilkanMahasiswa() {
    if (jumlahMahasiswa == 0) {
        cout << "Belum ada data mahasiswa.\n";
        return;
    }
    cout << "Data Mahasiswa:\n";
    for (int i = 0; i < jumlahMahasiswa; i++) {
        if (dataMahasiswa[i] != nullptr) {
```

```
            cout << "NIM: " << dataMahasiswa[i]->NIM
                 << ", Nama: " << dataMahasiswa[i]->nama
                 << ", IPK: " << dataMahasiswa[i]->IPK << endl;
        }
    }
}

// Fungsi untuk mengurutkan data mahasiswa berdasarkan IPK
menggunakan Bubble Sort
void urutkanMahasiswaByIPK() {
    for (int i = 0; i < jumlahMahasiswa - 1; i++) {
        for (int j = 0; j < jumlahMahasiswa - i - 1; j++) {
            if (dataMahasiswa[j]->IPK < dataMahasiswa[j + 1]->IPK) {
                Mahasiswa* temp = dataMahasiswa[j];
                dataMahasiswa[j] = dataMahasiswa[j + 1];
                dataMahasiswa[j + 1] = temp;
            }
        }
    }
    cout << "Data mahasiswa berhasil diurutkan berdasarkan IPK
(terbesar ke terkecil).\n";
}

// Fungsi utama
int main() {
    int pilihan;
    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Mahasiswa\n";
        cout << "2. Hapus Mahasiswa\n";
        cout << "3. Tampilkan Mahasiswa\n";
        cout << "4. Urutkan Mahasiswa berdasarkan IPK\n";
        cout << "5. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                tambahMahasiswa();
                break;
            case 2:
                hapusMahasiswa();
                break;
            case 3:
                tampilkanMahasiswa();
                break;
            case 4:
                urutkanMahasiswaByIPK();
                break;
            case 5:
                cout << "Keluar dari program.\n";
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 5);

    // Menghapus data mahasiswa yang tersisa untuk menghindari
memory leak
```

```
    for (int i = 0; i < jumlahMahasiswa; i++) {
        delete dataMahasiswa[i];
    }
    return 0;
}
```

Tampilan program



2. Soal Struct dan File Handling

Penjelasan program yang digunakan

1. Struct Peralatan: Terdiri dari kode, nama, jumlah, dan kondisi.
2. Fungsi tambahPeralatan: Menambah peralatan baru dan menyimpannya ke file.
3. Fungsi bacaData: Membaca data peralatan dari file dan menyimpannya ke dalam vector.
4. Fungsi tampilkanPeralatan: Menampilkan semua data peralatan.
5. Fungsi hapusPeralatan: Menghapus data peralatan berdasarkan kode.
6. Fungsi ubahPeralatan: Mengubah data peralatan berdasarkan kode.
7. Fungsi urutkanPeralatan: Mengurutkan peralatan berdasarkan kode menggunakan sort.
8. Main Menu: Menampilkan menu pilihan dan menerima input pengguna.

Cara Menggunakan Program

1. Gunakan Tambah Peralatan untuk memasukkan data.
2. Pilih Ubah Peralatan atau Hapus Peralatan untuk mengubah atau menghapus data.
3. Gunakan Tampilkan Peralatan untuk melihat semua data peralatan.
4. Pilih Urutkan Peralatan berdasarkan Kode untuk menampilkan data yang diurutkan.

Codingan

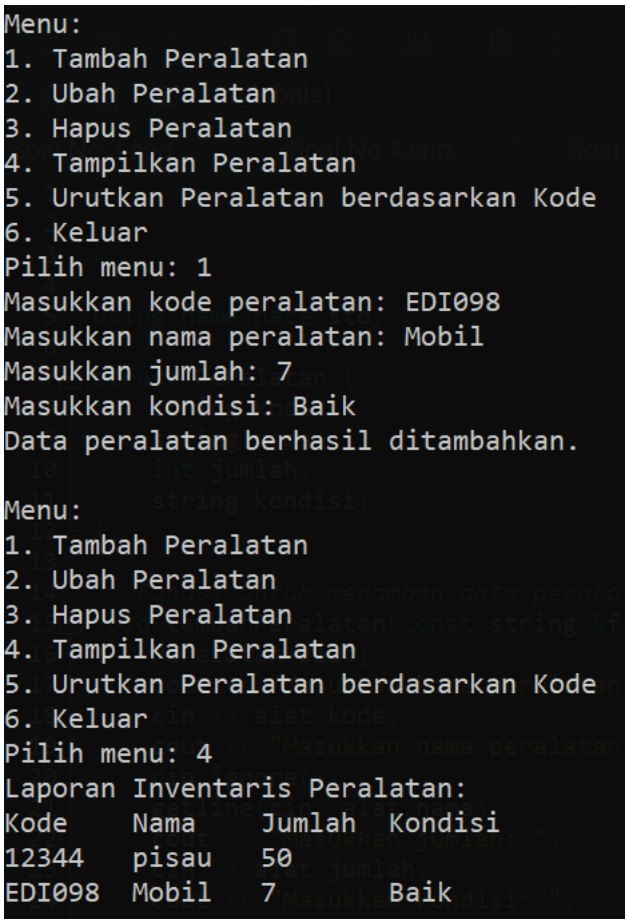
```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
using namespace std;

struct Peralatan {
    string kode;
    string nama;
    int jumlah;
    string kondisi;
};

// Fungsi untuk menambah data peralatan ke dalam file
void tambahPeralatan(const string &filename) {
    Peralatan alat;
    cout << "Masukkan kode peralatan: ";
    cin >> alat.kode;
    cout << "Masukkan nama peralatan: ";
    cin.ignore();
    getline(cin, alat.nama);
    cout << "Masukkan jumlah: ";
```

```
    cin >> alat.jumlah;
    cout << "Masukkan kondisi: ";
    cin.ignore();
    getline(cin, alat.kondisi);

    ofstream file(filename, ios::app);
    if (file.is_open()) {
        file << alat.kode << "," << alat.nama << "," << alat.jumlah
<< "," << alat.kondisi << endl;
        file.close();
        cout << "Data peralatan berhasil ditambahkan.\n";
    } else {
        cout << "Gagal membuka file.\n";
    }
}

// Fungsi untuk membaca data dari file
vector<Peralatan> bacaData(const string &filename) {
    vector<Peralatan> daftarAlat;
    ifstream file(filename);
    if (file.is_open()) {
        Peralatan alat;
        while (getline(file, alat.kode, ',') &&
               getline(file, alat.nama, ',') &&
               file >> alat.jumlah &&
               file.ignore() &&
               getline(file, alat.kondisi)) {
            daftarAlat.push_back(alat);
        }
        file.close();
    }
    return daftarAlat;
}

// Fungsi untuk menampilkan semua data peralatan
void tampilkanPeralatan(const vector<Peralatan> &daftarAlat) {
    if (daftarAlat.empty()) {
        cout << "Tidak ada data peralatan.\n";
        return;
    }

    cout << "Laporan Inventaris Peralatan:\n";
    cout << "Kode\tNama\tJumlah\tKondisi\n";
    for (const auto &alat : daftarAlat) {
        cout << alat.kode << "\t" << alat.nama << "\t" <<
alat.jumlah << "\t" << alat.kondisi << endl;
    }
}

// Fungsi untuk menghapus data peralatan berdasarkan kode
void hapusPeralatan(const string &filename, const string &kode) {
    vector<Peralatan> daftarAlat = bacaData(filename);
    auto it = remove_if(daftarAlat.begin(), daftarAlat.end(),
[&](const Peralatan &alat) {
        return alat.kode == kode;
    });

    if (it == daftarAlat.end()) {
```

```
        cout << "Peralatan dengan kode " << kode << " tidak
ditemukan.\n";
        return;
    }

    daftarAlat.erase(it, daftarAlat.end());
    ofstream file(filename, ios::trunc);
    for (const auto &alat : daftarAlat) {
        file << alat.kode << "," << alat.nama << "," << alat.jumlah
<< "," << alat.kondisi << endl;
    }
    file.close();
    cout << "Data peralatan berhasil dihapus.\n";
}

// Fungsi untuk mengubah data peralatan berdasarkan kode
void ubahPeralatan(const string &filename, const string &kode) {
    vector<Peralatan> daftarAlat = bacaData(filename);
    bool ditemukan = false;

    for (auto &alat : daftarAlat) {
        if (alat.kode == kode) {
            cout << "Data lama - Nama: " << alat.nama << ", Jumlah:
" << alat.jumlah << ", Kondisi: " << alat.kondisi << endl;
            cout << "Masukkan data baru:\n";
            cout << "Nama peralatan: ";
            cin.ignore();
            getline(cin, alat.nama);
            cout << "Jumlah: ";
            cin >> alat.jumlah;
            cout << "Kondisi: ";
            cin.ignore();
            getline(cin, alat.kondisi);
            ditemukan = true;
            break;
        }
    }

    if (!ditemukan) {
        cout << "Peralatan dengan kode " << kode << " tidak
ditemukan.\n";
        return;
    }

    ofstream file(filename, ios::trunc);
    for (const auto &alat : daftarAlat) {
        file << alat.kode << "," << alat.nama << "," << alat.jumlah
<< "," << alat.kondisi << endl;
    }
    file.close();
    cout << "Data peralatan berhasil diubah.\n";
}

// Fungsi untuk mengurutkan dan menampilkan data peralatan
berdasarkan kode
void urutkanPeralatan(vector<Peralatan> &daftarAlat) {
    sort(daftarAlat.begin(), daftarAlat.end(), [](const Peralatan
&a, const Peralatan &b) {
        return a.kode < b.kode;
```

```
    });
    tampilkanPeralatan(daftarAlat);
}

int main() {
    string filename = "inventaris.txt";
    int pilihan;
    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Peralatan\n";
        cout << "2. Ubah Peralatan\n";
        cout << "3. Hapus Peralatan\n";
        cout << "4. Tampilkan Peralatan\n";
        cout << "5. Urutkan Peralatan berdasarkan Kode\n";
        cout << "6. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                tambahPeralatan(filename);
                break;
            case 2: {
                string kode;
                cout << "Masukkan kode peralatan yang ingin diubah:
";
                cin >> kode;
                ubahPeralatan(filename, kode);
                break;
            }
            case 3: {
                string kode;
                cout << "Masukkan kode peralatan yang ingin dihapus:
";
                cin >> kode;
                hapusPeralatan(filename, kode);
                break;
            }
            case 4: {
                vector<Peralatan> daftarAlat = bacaData(filename);
                tampilkanPeralatan(daftarAlat);
                break;
            }
            case 5: {
                vector<Peralatan> daftarAlat = bacaData(filename);
                urutkanPeralatan(daftarAlat);
                break;
            }
            case 6:
                cout << "Keluar dari program.\n";
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 6);

    return 0;
}
```

Tampilan program



3. Soal stack

Penjelesan program yang digunakan

1. Class Stack:

- Mempunyai fungsi push untuk menambahkan elemen, pop untuk mengeluarkan elemen dari stack, dan isEmpty untuk memeriksa apakah stack kosong.
- top digunakan untuk melihat elemen teratas di stack.

2. Fungsi evaluatePostfix:

- Membaca setiap token dari ekspresi postfix.
- Jika token berupa angka, ia menambahkannya ke stack.
- Jika token adalah operator, ia mengeluarkan dua angka teratas dari stack, menerapkan operator tersebut, lalu menambahkan hasilnya kembali ke stack.
- Mendukung operator +, -, *, /, dan ^ (pangkat).
- Setiap langkah evaluasi dicetak ke layar untuk menunjukkan proses yang terjadi.

3. Main Function:

- Membaca ekspresi postfix dari input pengguna, mengevaluasinya, dan menampilkan hasilnya.

Codingan

```
#include <iostream>
#include <stack>
#include <string>
#include <cmath>
#include <sstream>
#include <vector>
using namespace std;

class Stack {
private:
    vector<double> elements;
public:
    void push(double value) {
        elements.push_back(value);
    }

    double pop() {
        if (isEmpty()) {
            throw runtime_error("Stack is empty!");
        }
        double value = elements.back();
        elements.pop_back();
        return value;
    }

    bool isEmpty() const {
        return elements.empty();
    }

    double top() const {
        if (isEmpty()) {
            throw runtime_error("Stack is empty!");
        }
        return elements.back();
    }
};

double evaluatePostfix(const string &expression) {
    Stack stack;
    istringstream iss(expression);
    string token;

    cout << "Langkah-langkah evaluasi ekspresi postfix:\n";

    while (iss >> token) {

        if (isdigit(token[0]) || (token[0] == '-' && token.length()
> 1)) {
            double number = stod(token);
            stack.push(number);
```

```
            cout << "Push " << number << " ke stack\n";
        }

        else {
            double operand2 = stack.pop();
            double operand1 = stack.pop();
            double result;

            if (token == "+") {
                result = operand1 + operand2;
            } else if (token == "-") {
                result = operand1 - operand2;
            } else if (token == "*") {
                result = operand1 * operand2;
            } else if (token == "/") {
                result = operand1 / operand2;
            } else if (token == "^") {
                result = pow(operand1, operand2);
            } else {
                throw runtime_error("Unknown operator: " + token);
            }

            stack.push(result);
            cout << "Pop " << operand1 << " dan " << operand2 << ",
lalu " << operand1 << " " << token << " " << operand2 << " = " <<
result << "\n";
        }
    }

    double finalResult = stack.pop();
    if (!stack.isEmpty()) {
        throw runtime_error("Ekspresi postfix tidak valid.");
    }
    return finalResult;
}

int main() {
    string expression;
    cout << "Masukkan ekspresi postfix (contoh: '3 4 + 2 * 7 /'): ";
    getline(cin, expression);

    try {
        double result = evaluatePostfix(expression);
        cout << "Hasil dari ekspresi postfix adalah: " << result <<
endl;
    } catch (const exception &e) {
        cout << "Error: " << e.what() << endl;
    }

    return 0;
}
```

Tampilan program

```
Masukkan ekspresi postfix (contoh: '3 4 + 2 * 7 /'): 2+3
Langkah-langkah evaluasi ekspresi postfix:
Push 2 ke stack
Hasil dari ekspresi postfix adalah: 2
```

4. Soal queue

1. **Struct Pelanggan**:

- Memiliki atribut nomorAntrian (nomor pelanggan) dan waktuLayanan (waktu layanan dalam menit).

2. **Kelas Queue**:

- Memiliki fungsi enqueue untuk menambahkan pelanggan ke dalam antrian.
- dequeue untuk menghapus pelanggan dari antrian.
- isEmpty untuk memeriksa apakah antrian kosong, dan size untuk mendapatkan jumlah pelanggan dalam antrian.

3. **Fungsi simulasiLayanan**:

- Menjalankan simulasi dengan jumlah pelanggan dan loket yang ditentukan.
- Setiap pelanggan memiliki waktu layanan acak antara 1 hingga 10 menit.
- Loket melayani pelanggan secara paralel. Jika sebuah loket selesai melayani pelanggan, loket akan mengambil pelanggan berikutnya dari antrian.
- Menghitung statistik akhir termasuk rata-rata waktu tunggu, total pelanggan yang terlayani, dan sisa antrian.

4. **Main Function**:

- Meminta input jumlah pelanggan dari pengguna dan menjalankan simulasi antrian layanan.

Codingan

```
#include <iostream>
#include <queue>
#include <cstdlib>
#include <ctime>
#include <vector>
using namespace std;

// Struktur Pelanggan dengan nomor antrian dan waktu layanan
struct Pelanggan {
    int nomorAntrian;
    int waktuLayanan; // waktu layanan dalam menit
};
```

```
// Kelas Queue untuk operasi antrian
class Queue {
private:
    queue<Pelanggan> q;

public:
    void enqueue(const Pelanggan &p) {
        q.push(p);
    }

    Pelanggan dequeue() {
        if (isEmpty()) {
            throw runtime_error("Queue kosong!");
        }
        Pelanggan front = q.front();
        q.pop();
        return front;
    }

    bool isEmpty() const {
        return q.empty();
    }

    int size() const {
        return q.size();
    }
};

// Fungsi untuk menghitung statistik antrian layanan
void simulasiLayanan(int jumlahPelanggan, int jumlahLoket) {
    Queue antrian;
    vector<int> waktuLoket(jumlahLoket, 0);  // waktu layanan per
loket
    int totalWaktuTunggu = 0;
    int totalPelangganTerlayani = 0;

    // Memasukkan pelanggan ke dalam antrian dengan nomor dan waktu
layanan acak
    for (int i = 1; i <= jumlahPelanggan; i++) {
        int waktuLayanan = rand() % 10 + 1; // waktu layanan acak
antara 1-10 menit
        antrian.enqueue({i, waktuLayanan});
    }

    // Memproses antrian pelanggan dengan simulasi per menit
    while (!antrian.isEmpty()) {
        for (int i = 0; i < jumlahLoket; i++) {
            if (waktuLoket[i] == 0 && !antrian.isEmpty()) {
                Pelanggan pelanggan = antrian.dequeue();
                waktuLoket[i] = pelanggan.waktuLayanan;
                totalWaktuTunggu += pelanggan.waktuLayanan;
                totalPelangganTerlayani++;
                cout << "Pelanggan #" << pelanggan.nomorAntrian << "
dilayani di loket " << i + 1
                     << " dengan waktu layanan " <<
pelanggan.waktuLayanan << " menit.\n";
            }
        }
```

```
        // Mengurangi waktu layanan setiap loket per menit
        for (int i = 0; i < jumlahLoket; i++) {
            if (waktuLoket[i] > 0) {
                waktuLoket[i]--;
            }
        }
    }

    // Menghitung statistik akhir
    double rataRataWaktuTunggu = totalWaktuTunggu /
(double)totalPelangganTerlayani;
    cout << "\nStatistik Layanan:\n";
    cout << "Rata-rata waktu tunggu: " << rataRataWaktuTunggu << "
menit\n";
    cout << "Total pelanggan terlayani: " << totalPelangganTerlayani
<< endl;
    cout << "Sisa antrian: " << antrian.size() << endl;
}

int main() {
    srand(time(0)); // Mengatur seed untuk waktu layanan acak
    int jumlahPelanggan, jumlahLoket = 3;

    cout << "Masukkan jumlah pelanggan yang akan dilayani: ";
    cin >> jumlahPelanggan;

    simulasiLayanan(jumlahPelanggan, jumlahLoket);

    return 0;
}
```

Tampilan program

```
Masukkan jumlah pelanggan yang akan dilayani: 4
Pelanggan #1 dilayani di loket 1 dengan waktu layanan 2 menit.
Pelanggan #2 dilayani di loket 2 dengan waktu layanan 7 menit.
Pelanggan #3 dilayani di loket 3 dengan waktu layanan 1 menit.
Pelanggan #4 dilayani di loket 3 dengan waktu layanan 7 menit.

Statistik Layanan:
Rata-rata waktu tunggu: 4.25 menit
Total pelanggan terlayani: 4
Sisa antrian: 0
```

5. Soal Implementasi Gabungan

1. **Struct Buku**:
   - Mempunyai atribut ISBN, judul, pengarang, dan tahunTerbit.
2. **Array daftarBuku**:
   - Menyimpan pointer ke objek Buku untuk menyimpan data buku perpustakaan.
3. **Queue antrianPeminjaman**:

- Mengelola antrian peminjaman buku menggunakan struktur queue.

4. **Stack riwayatPeminjaman**:
   - Menyimpan riwayat buku yang telah dipinjam menggunakan struktur stack.
5. **Fungsi-fungsi Utama**:
   - tambahBuku: Menambahkan data buku baru ke perpustakaan.
   - cariBuku: Mencari buku berdasarkan ISBN.
   - tampilkanBuku: Menampilkan semua buku yang tersedia di perpustakaan.
   - pinjamBuku: Menambahkan buku yang akan dipinjam ke antrian.
   - prosesPeminjaman: Memproses buku dari antrian dan menambahkannya ke riwayat peminjaman.
   - kembalikanBuku: Mengembalikan buku dari stack riwayat peminjaman.
   - tampilkanRiwayatPeminjaman: Menampilkan riwayat buku yang telah dipinjam.
6. **Main Function**:
   - Menampilkan menu interaktif untuk mengelola perpustakaan.

Codingan

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

struct Buku {
    string ISBN;
    string judul;
    string pengarang;
    int tahunTerbit;
};

const int MAX_BUKU = 100;
Buku* daftarBuku[MAX_BUKU];
int jumlahBuku = 0;

stack<Buku*> riwayatPeminjaman;

queue<Buku*> antrianPeminjaman;

void tambahBuku() {
    if (jumlahBuku >= MAX_BUKU) {
        cout << "Kapasitas perpustakaan penuh!\n";
        return;
    }

    Buku* bukuBaru = new Buku;
    cout << "Masukkan ISBN: ";
```

```
    cin >> bukuBaru->ISBN;
    cout << "Masukkan judul buku: ";
    cin.ignore();
    getline(cin, bukuBaru->judul);
    cout << "Masukkan pengarang: ";
    getline(cin, bukuBaru->pengarang);
    cout << "Masukkan tahun terbit: ";
    cin >> bukuBaru->tahunTerbit;

    daftarBuku[jumlahBuku++] = bukuBaru;
    cout << "Buku berhasil ditambahkan.\n";
}

Buku* cariBuku(const string& ISBN) {
    for (int i = 0; i < jumlahBuku; i++) {
        if (daftarBuku[i]->ISBN == ISBN) {
            return daftarBuku[i];
        }
    }
    return nullptr;
}

void tampilkanBuku() {
    if (jumlahBuku == 0) {
        cout << "Tidak ada buku di perpustakaan.\n";
        return;
    }

    cout << "Daftar Buku:\n";
    for (int i = 0; i < jumlahBuku; i++) {
        cout << "ISBN: " << daftarBuku[i]->ISBN
             << ", Judul: " << daftarBuku[i]->judul
             << ", Pengarang: " << daftarBuku[i]->pengarang
             << ", Tahun Terbit: " << daftarBuku[i]->tahunTerbit <<
endl;
    }
}

void pinjamBuku() {
    string ISBN;
    cout << "Masukkan ISBN buku yang ingin dipinjam: ";
    cin >> ISBN;

    Buku* buku = cariBuku(ISBN);
    if (buku) {
        antrianPeminjaman.push(buku);
        cout << "Buku dengan judul '" << buku->judul << "'
ditambahkan ke antrian peminjaman.\n";
    } else {
        cout << "Buku dengan ISBN " << ISBN << " tidak
ditemukan.\n";
    }
}

void prosesPeminjaman() {
    if (antrianPeminjaman.empty()) {
        cout << "Tidak ada peminjaman dalam antrian.\n";
        return;
    }
```

```
    Buku* bukuDipinjam = antrianPeminjaman.front();
    antrianPeminjaman.pop();
    riwayatPeminjaman.push(bukuDipinjam);
    cout << "Buku dengan judul '" << bukuDipinjam->judul << "' telah
dipinjam.\n";
}

void kembalikanBuku() {
    if (riwayatPeminjaman.empty()) {
        cout << "Tidak ada buku yang perlu dikembalikan.\n";
        return;
    }

    Buku* bukuDikembalikan = riwayatPeminjaman.top();
    riwayatPeminjaman.pop();
    cout << "Buku dengan judul '" << bukuDikembalikan->judul << "'
telah dikembalikan.\n";
}

void tampilkanRiwayatPeminjaman() {
    if (riwayatPeminjaman.empty()) {
        cout << "Tidak ada riwayat peminjaman.\n";
        return;
    }

    stack<Buku*> tempRiwayat = riwayatPeminjaman;
    cout << "Riwayat Peminjaman:\n";
    while (!tempRiwayat.empty()) {
        Buku* buku = tempRiwayat.top();
        cout << "Judul: " << buku->judul << ", ISBN: " << buku->ISBN
<< endl;
        tempRiwayat.pop();
    }
}

int main() {
    int pilihan;
    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Buku\n";
        cout << "2. Cari Buku\n";
        cout << "3. Tampilkan Semua Buku\n";
        cout << "4. Pinjam Buku\n";
        cout << "5. Proses Antrian Peminjaman\n";
        cout << "6. Kembalikan Buku\n";
        cout << "7. Tampilkan Riwayat Peminjaman\n";
        cout << "8. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                tambahBuku();
                break;
            case 2: {
                string ISBN;
                cout << "Masukkan ISBN buku yang dicari: ";
                cin >> ISBN;
```

```
                Buku* buku = cariBuku(ISBN);
                if (buku) {
                    cout << "ISBN: " << buku->ISBN
                         << ", Judul: " << buku->judul
                         << ", Pengarang: " << buku->pengarang
                         << ", Tahun Terbit: " << buku->tahunTerbit
<< endl;
                } else {
                    cout << "Buku tidak ditemukan.\n";
                }
                break;
            }
            case 3:
                tampilkanBuku();
                break;
            case 4:
                pinjamBuku();
                break;
            case 5:
                prosesPeminjaman();
                break;
            case 6:
                kembalikanBuku();
                break;
            case 7:
                tampilkanRiwayatPeminjaman();
                break;
            case 8:
                cout << "Keluar dari program.\n";
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 8);

    for (int i = 0; i < jumlahBuku; i++) {
        delete daftarBuku[i];
    }

    return 0;
}
```

Tampilan program