**SISTEM PEMBELIAN TIKET BIOSKOP**

Diajukan Sebagai Tugas Mata Kuliah

Struktur Data



**Anggota Kelompok :**

|  |  |
| --- | --- |
| **Albertus Ivan Suryawan** | **2008561093** |
| **I Gusti Bgs Darmika Putra** | **2008561094** |
| **I Wayan Agus Juniartha** | **2008561095** |
| **Dewa Nyoman Agung Adipurwa Mahandiri** | **2008561095** |

**PRODI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**BUKIT JIMBARAN**

**20**

**Daftar Isi**

**Halaman Judul**

**Daftar Isii**

**1. Pendahuluan1**

1.1 Lata Belakang1

1.2 Rumusan Masalah1

1.3 Tujuan1

**2. Analisi dan Rancangan2**

2.1 Analisi Masalah2

2.2 Analisi Struktur Data2

2.3 Analisi Algoritma3

2.4 Rancangan Flowcart6

**3. Implementasi**14

3.1 Source Code14

3.2 Output25

**4. Kesimpulan28**

# **Pendahuluan**

## Latar Belakang

Pada dunia programmer ada banyak jenis-jenis bahasa yang dapat digunakan untuk membangun suatu program.Bahasa pemrograman merupakan kumpulan aturan yang disusun sedemikian rupa sehingga memungkinkan pengguna komputer membuat program yang dapat dijalankan dengan aturan tersebut.Bahasa pemrograman dapat dikelompokkan dalam berbagai macam sudut pandang. Penerapan pemrograman dalam dunia bisnis dapat dilihat seperti program untuk sistem pembelian tiket bioskop yang merupakan salah satu solusi agar meminimalisir antrian yang panjang ketika membeli tiket secara langsung, saat ini pemesanan tiket di bioskop dapat dilakukan secara online atau melelui halaman web bioskop tersebut. Pemesanan tiket di bioskop menggunakan halaman web itu sangat memudahkan customer dalam memesan tiket bioskop. Dengan cara ini maka customer dapat memilih bioskop,film, jam tayang, dan tempat duduk,yang diinginkanya sesui kebutuhan customer.

## Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah yang diambil dalam penulisan laporan ini adalah:

1. Apa saja konsep-konsep yang diperlukan?
2. Tipe data apa saja yang digunakan pada program ini?
3. Bagaimana alur dari program ini?
4. Bagaimana implementasikan dari sistem bioskop?
   1. Tujuan

Membangun sebuah sistem pemesanan tiket bertujuan untuk mempermudah pembeli untuk melakukan pemesan tiket dan untuk meminimalisir antrian yang panjang ketika membeli tiket secara langsung.

1. **Analisis dan Rancangan**
   1. Analisis Masalah

Antrian pada sistem pembelian tiket bioskop menangani pembayaran dan pengembalian sisa uang tiket dengan pecahannya dan penjadwalan film yang tayang, batasi jumlah studio bioskop hanya 3 buah dengan kapasitas jursi terbatas. Oleh karena itu, beberapa masalah yang dipecahkan dalam implentasi sistem bioskop antara lain yaitu servis pelanggan dan penjadwalan film.

* 1. Analisis Struktur Data

Konsep truktur data yang digunakan didalam proyek ini terbagi menjadi 3 bagian yaitu sistem antiran dan kasir dalam servis pelanggang dan sistem penjadwalan.

Sistem antrian menggunakan beberapa struktur data antara lain:

* 1. Queue
  2. Array
  3. Abstrack Data Type (ADT)
  4. Linked list

Sistem kasir menggunakan beberapa struktur data antara lain:

1. Stack
2. Array
3. Abstrak Data Type (ADT)
4. Linked list

Penjadwalan film menggunakan beberapa struktur data antara lain:

* 1. Graph
  2. Hashing
  3. Array
  4. Abstrak Data Type (ADT)
  5. Linked list
  6. Recursion
  7. Analisis Algoritma

**Algoritma** Mengantri.

1. Minta nama pelanggan lalu alokasikan ke penyimpanan antrian
2. Jika penyimpanan kosong, tampilkan ERROR
3. Duplikat identitas pelanggan dari inputan ke idententitas dalam queue, ubah alamat queue menjadi NULL node terakhir dalam linkedlist.
4. Jika setelah duplikat masih bernilai NULL, keluar dari sistem antrian.
5. Jika duplikat tidak bernilai NULL, alokasi informasi dan alamat linked pelangga ke linked list baru misal temp. Periksa apakah temp sudah berada pada baris terakhir untuk mengantri. Jika belum ulang periksa hingga sampai di linked list antrian terkahir dengan alamt memori NULL. Jika sudah bernilai NULL, masukan informasi ke dalam antrian.

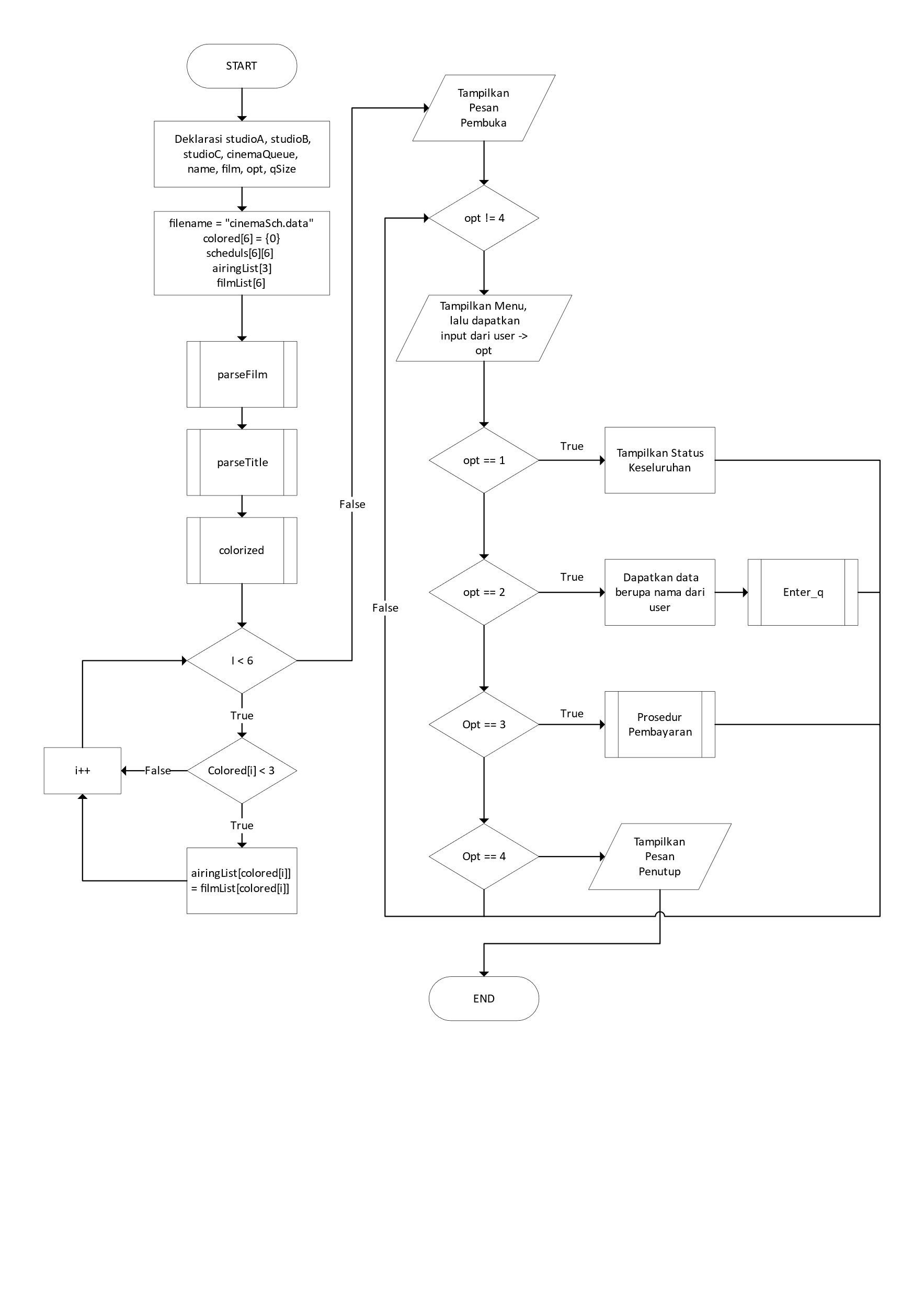
**Algoritma** Prosedur Pembayaran

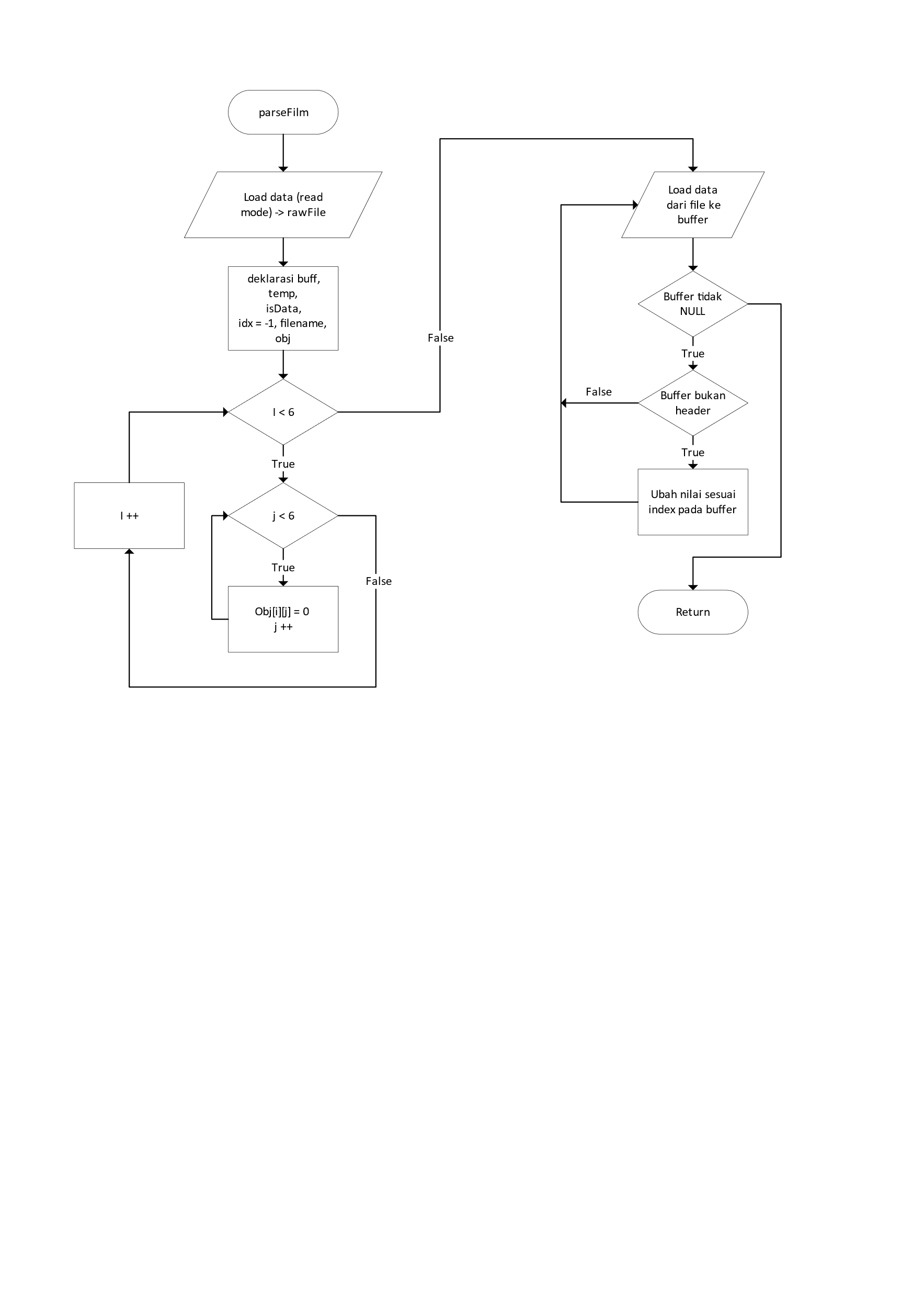
1. Keluarkan custumer paling depan dari queue, lalu duplikat nama pelanggan ke data pelanggan dalam prosedur pembayaran
2. Jika string nama bernilai NULL, tampilan ERROR lalu return
3. Jika false,tampilkan nama pelanggan, lalu tampilkan film yang tersedia.
4. Masukan inputan beripah pilihan film dari user.
5. Jika film yang tersedia belum tayang true, tampilkan peringatan apakah ingin melanjutkan transaki dan masukan pelanggan ke salah satu studio. Jika studio penuh lanjutkan ke studio lain, jika seluruh studio penuh beri peringatan kembali apakah ingin melanjutkan transaksi. Jika iya tampilkan kembali pilihan film dan lakukan hal yang sama, jika tidak keluar dari pemesanan tiket dan pembayaran batal.
6. Jika pelanggan berhasil mendapatkan studio yang tersedia, tambahkan pelanggan kedalam hashTable dari studio.
7. Tampilkan nama, judul film pilihan dan biaya. Masukan nomimal pembayaran dari user, Jika uang pembayaran kurang, ulang kembali proses. Jika tidak lanjutkan proses penghitungan dan beri kembalian jika ada.

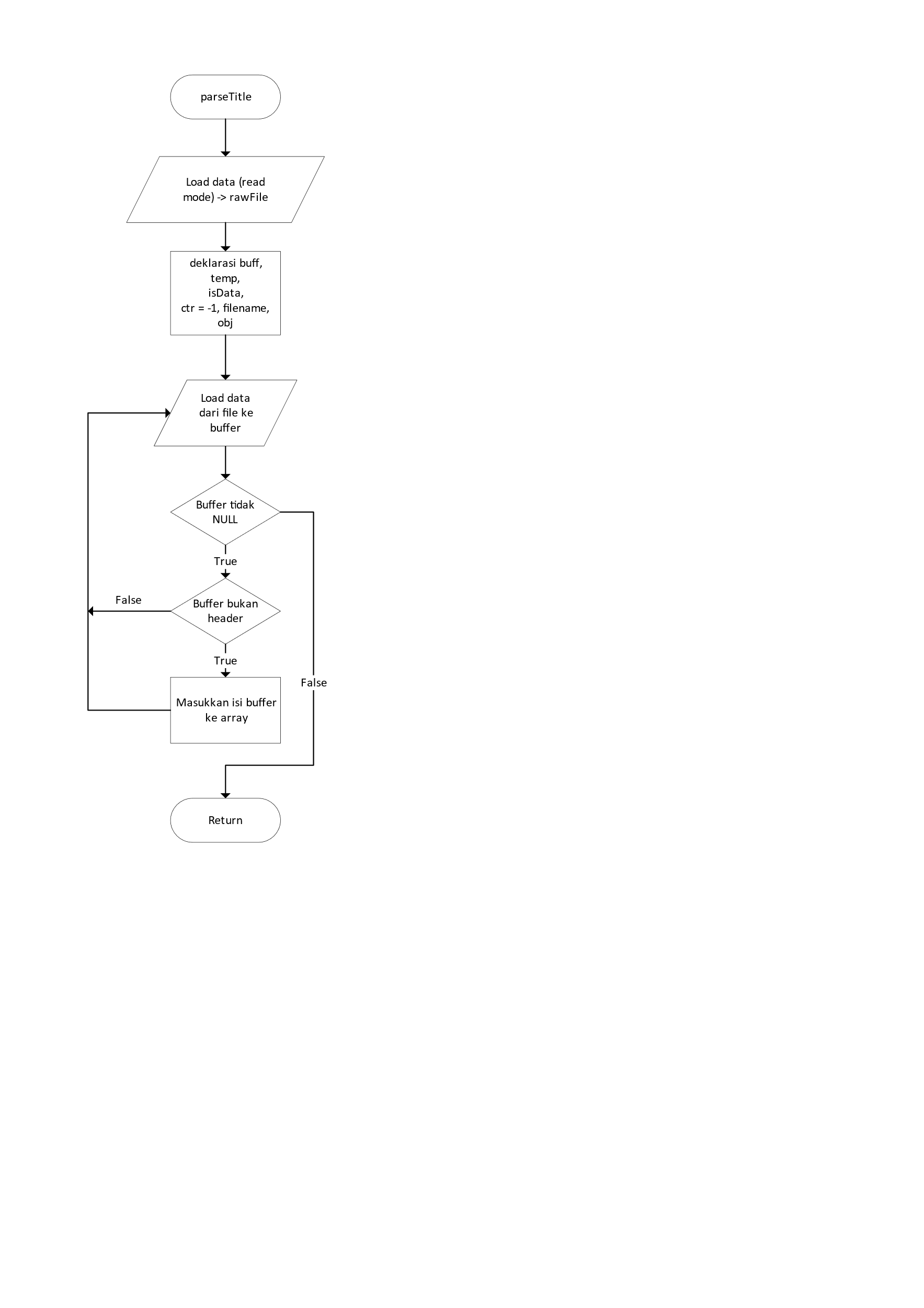
**Algoritma** Penjadwalan Film

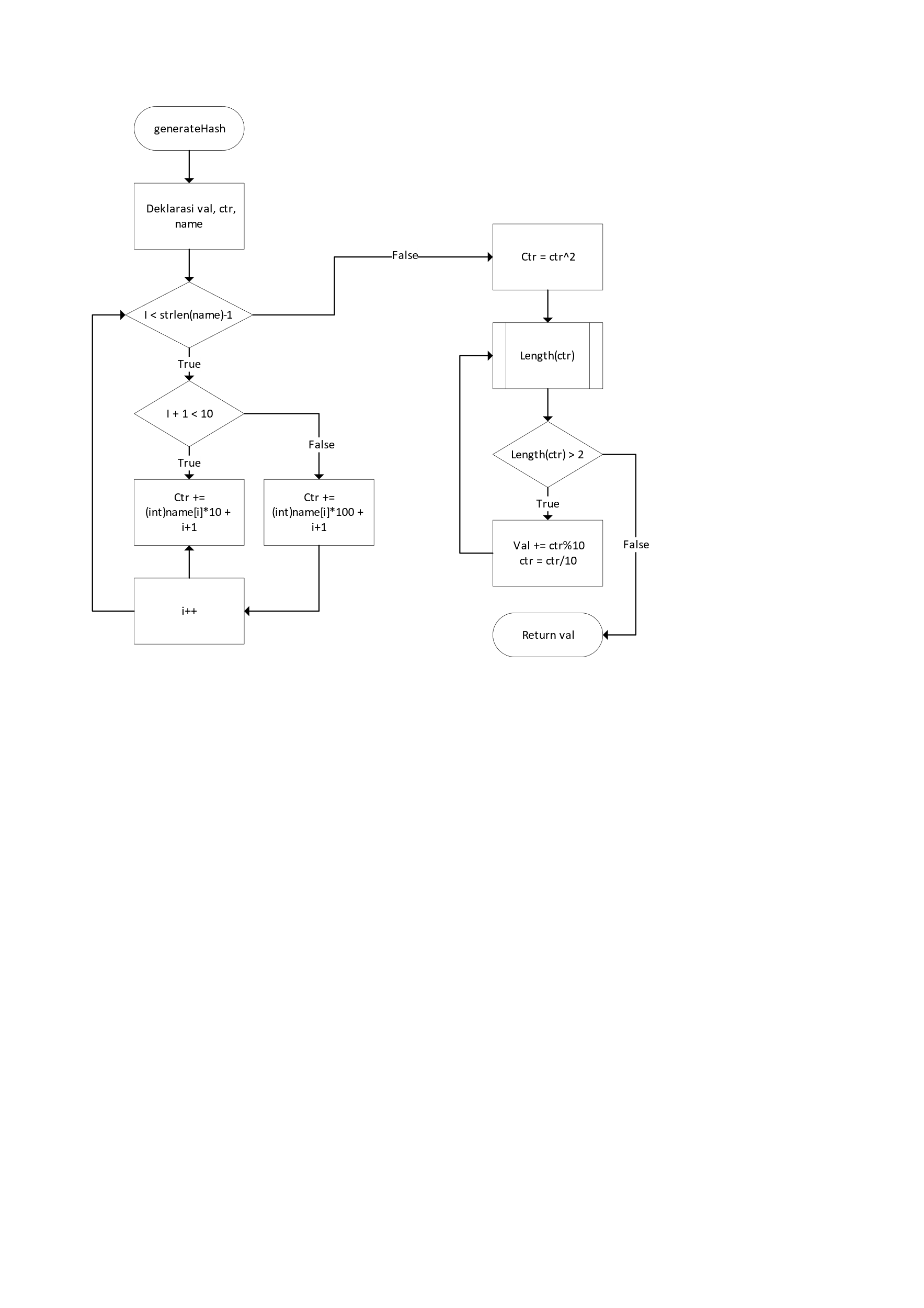
Algoritma ini terbagai atas memuat data judul, data film, dan pewarnaan graph

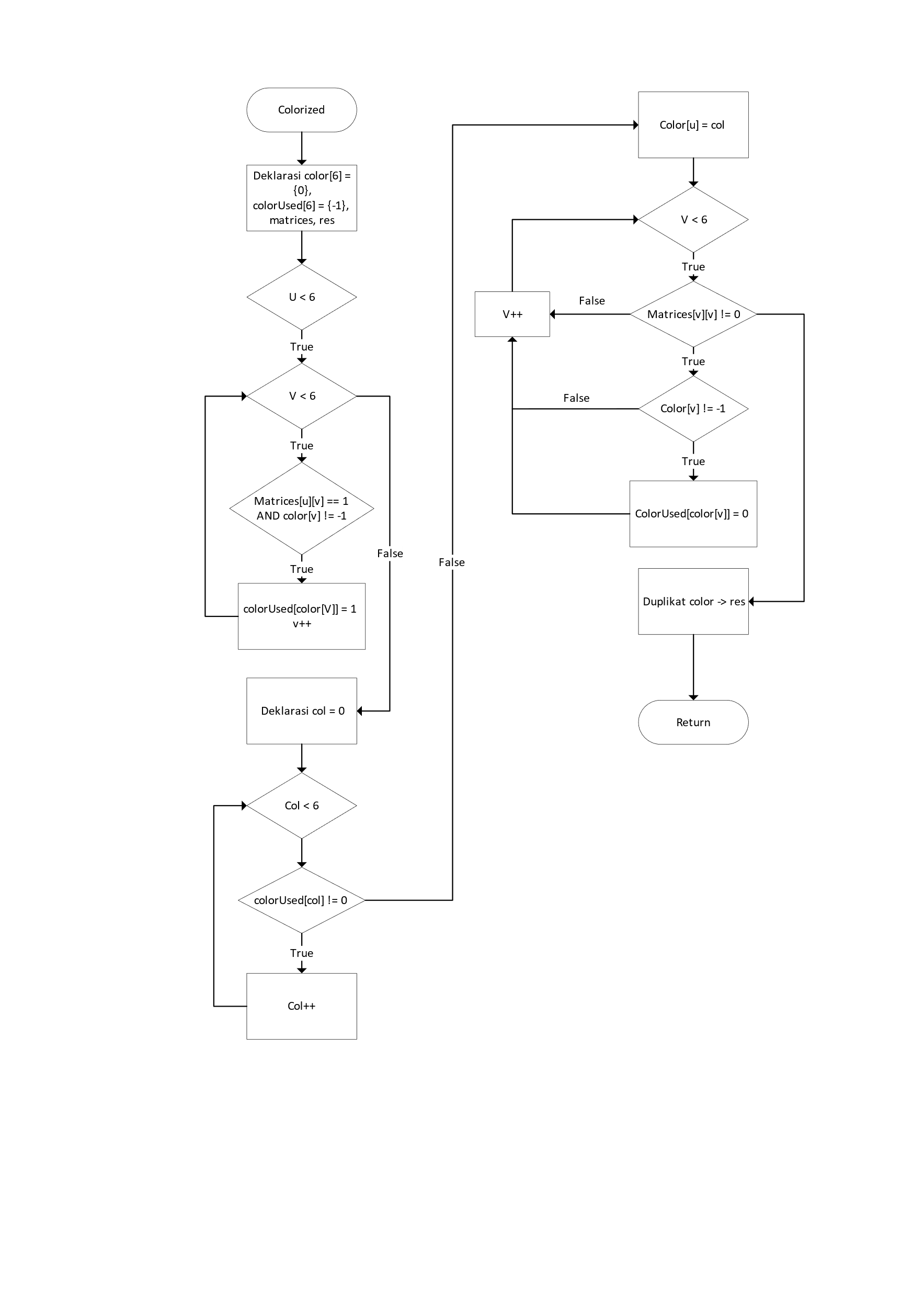
1. Buka file data film dan file data judul dari database
   1. Untuk data Film
      1. Alokasi data film dalam memori dan buffer
      2. Lakukan perulangan membaca matriks dua dimensi film dalm bentuk string untuk penjadwalan nantinya.
      3. Ubah data yang dibaca ke dalam bentuk integer dan alokasikan ke dalam matriks dua dimensi.
      4. Lakukan perulangan selama belum bernilai NULL
   2. Untuk data judul
      1. Alokasi data judul de dalam memori dan buffer
      2. Lakukan perulangan untuk membaca judul judul film dalam data.
      3. Inisiasi sebuah varibel bernilai true (1) jika pembacaan belum sampai apda akhir data
      4. Jika varibel berinilai true dan buff bernilai true duplikan judul film ke dalam array.
   3. Pewarnaan Graph
      1. Deklarasi sebuah varibel matriks dengan nilai 0, dan varibel matriks dengan nilai 1
      2. Lakukan perulangan untuk melakuakn pewarnaan
      3. Jika matriks dua dimensi dengan vertex identik (a,a) tidak berinilai 0 duplikan warna untuk di return.
   4. Rancangan Flowcart

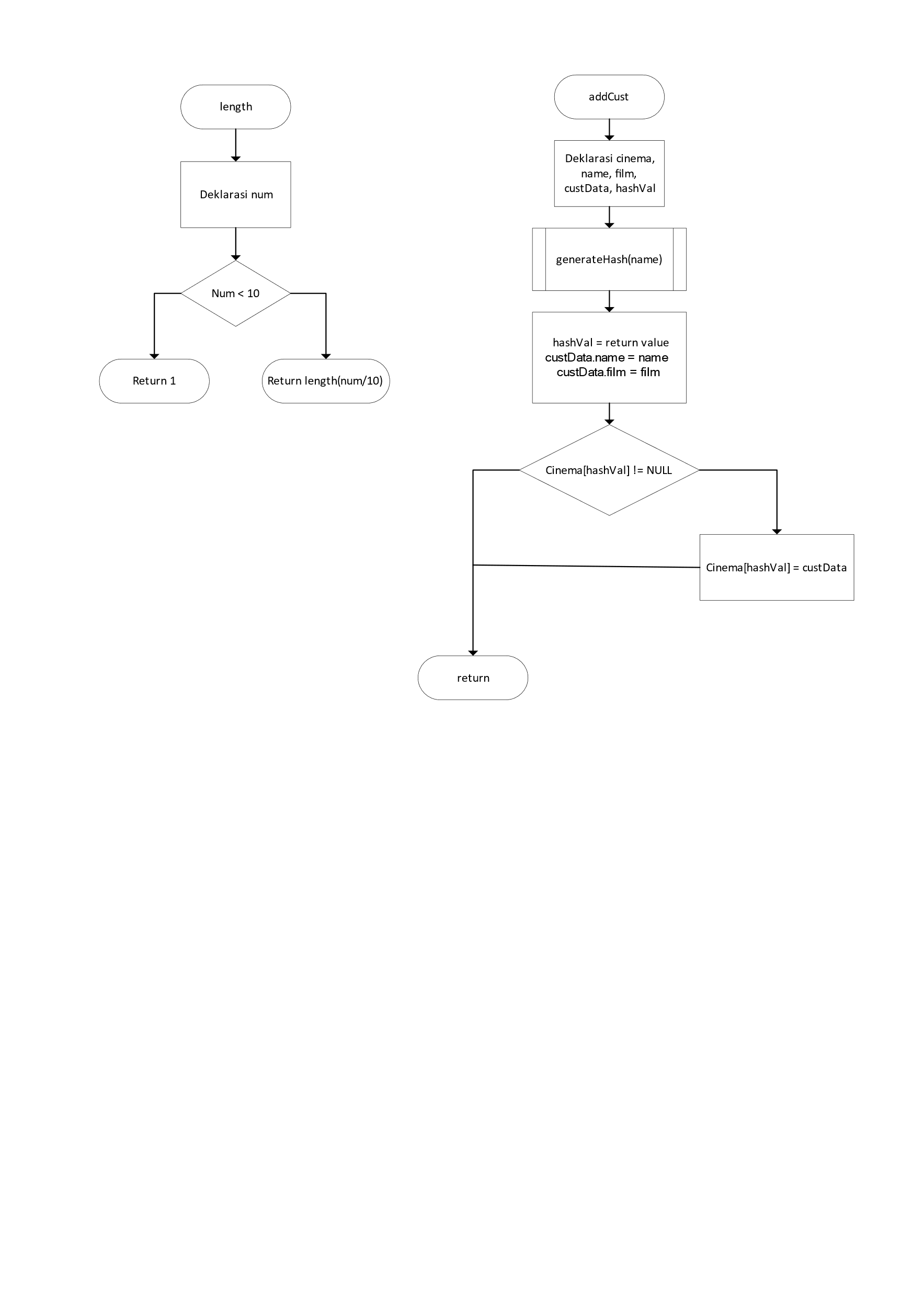
Adapaun alur flowcart dari program sistem pembelian tiket bioskop sebagai berikut

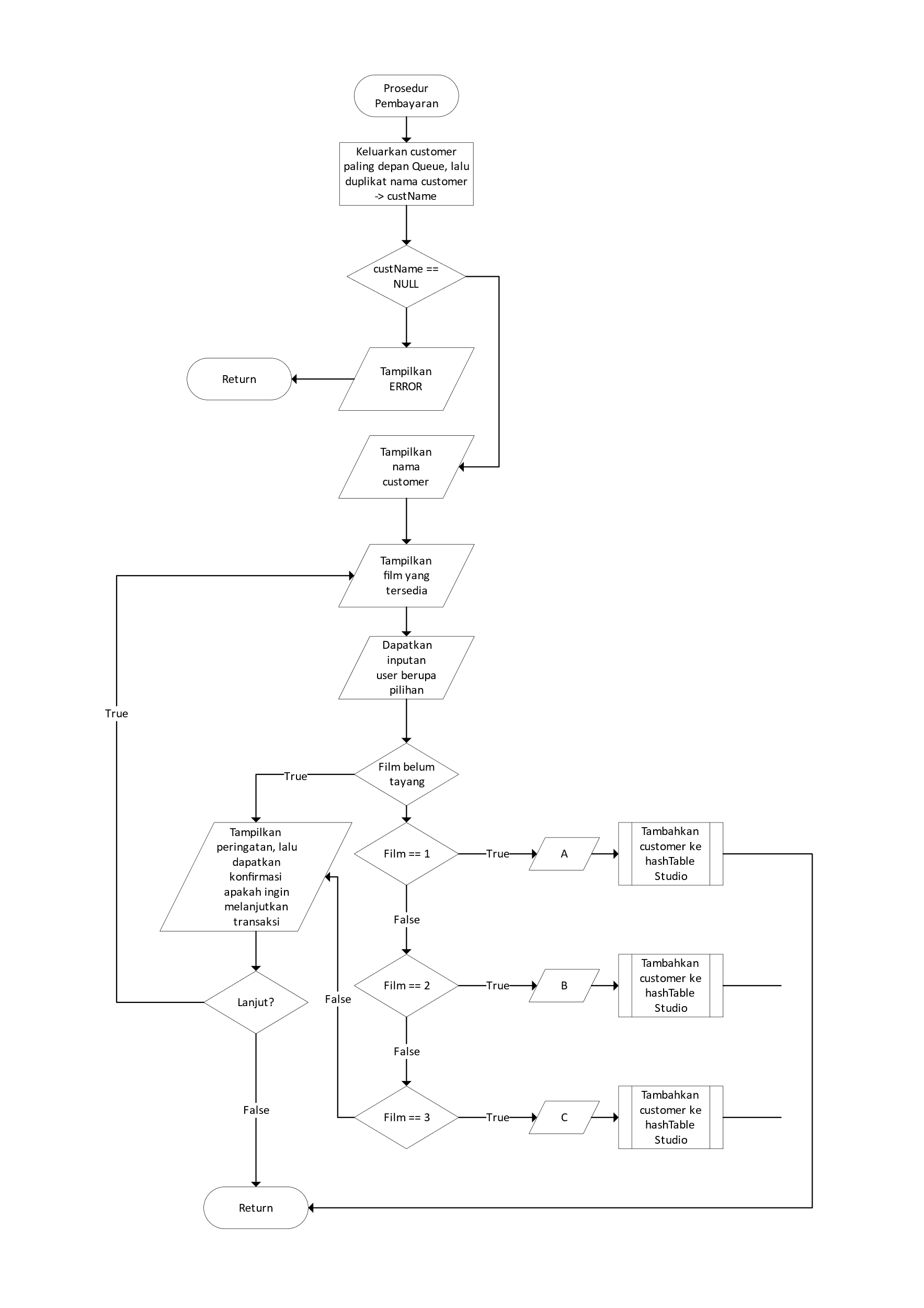
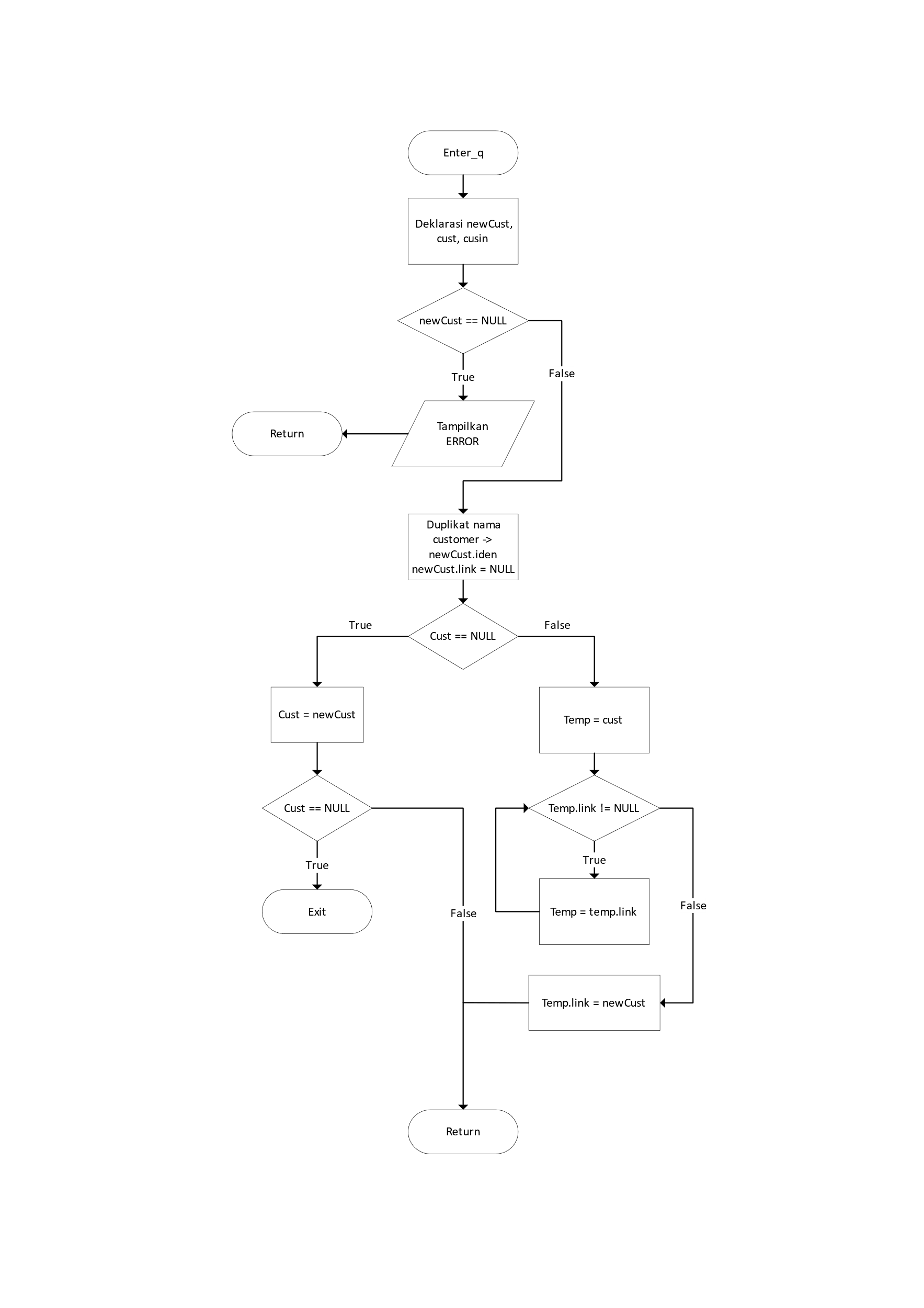












1. **Implementasi**
   1. **Source Code**

**Main: app.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <conio.h>

#include "libs\backend.h"

#include "libs\custServ.h"

*// void preload() {*

*//     FILE\* cache = fopen("queue.data", "r+");*

*// }*

*// void processPreload(queue cust, hash\* stdA, hash\* stdB, hash\* stdC) {*

*// }*

int main(int argc, char\* argv*[]*) {

    hash studioA[100];

    hash studioB[100];

    hash studioC[100];

    queue cinemaQueue = NULL;

    initCinema(studioA, studioB, studioC);

*// preload(cinemaQueue, studioA, studioB, studioC);*

    char\* name = calloc(255, sizeof(char));

    char\* film = calloc(255, sizeof(char));

    int opt = 0;

    int qSize = getQueueSize(cinemaQueue);

    printf("Selamat datang di Luminous Cinema!\n\n");

    while (opt!=4) {

        printf("[1] Lihat Antrian\n");

        printf("[2] Mengatri\n");

        printf("[3] Memesan\n");

        printf("[4] Keluar\n");

        printf("Input: ");

        scanf("%d", &opt);

        switch (opt) {

            case 1:

                printf("Jumlah Antrean %d!\n", getQueueSize(cinemaQueue));

                printf("Penonton pada Studio A: \n");

                getCinemaStats(studioA);

                printf("Penonton pada Studio B: \n");

                getCinemaStats(studioB);

                printf("Penonton pada Studio C: \n");

                getCinemaStats(studioC);

                printf("\n");

                break;

            case 2:

                printf("Masukkan Nama Anda: ");

                fflush(stdin);

                gets(name);

                enter\_q(&cinemaQueue, name);

                break;

            case 3:

                char\* custName = getCust(&cinemaQueue);

                if (custName == NULL) {

                    printf("Anda belum mengantri!\n");

                    break;

                }

                printf("Nama Customer: %s\n", custName);

                film = buyTicket(studioA, studioB, studioC, custName);

                if (film == NULL) break;

                getPayment(custName, film);

                break;

            case 4:

                printf("Terima kasih atas kunjungannya!\n");

                exit(0);

                break;

            default:

                printf("Masukkan pilihan yang sesuai!\n");

        }

        getch();

        system("cls");

    }

}

**Modular: backend.c**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#include <math.h>

#include "backend.h"

#include "custServ.h"

*static* char\* filmList[MAX\_NODE] = {0};

*static* char\* airingList[3] = {0};

*static* int schedules[MAX\_NODE][MAX\_NODE];

void initCinema(hash\* studioA, hash\* studioB, hash\* studioC) {

    char\* filename = "cinemaSch.data";

    int colored[MAX\_NODE] = {0};

    for (int i=0; i<MAX\_ARR; i++) {

        studioA[i] = NULL;

        studioB[i] = NULL;

        studioC[i] = NULL;

    }

    parseFilm(filename, schedules);

    parseTitle(filename, filmList);

    colorized(schedules, colored);

    for (int i=0; i<MAX\_NODE; i++) {

        if (colored[i] == 0 || colored[i] == 1 || colored[i] == 2) {

            if (!airingList[colored[i]]) {

                airingList[colored[i]] = strdup(filmList[colored[i]]);

            }

        }

    }

}

char\* buyTicket(hash\* studioA, hash\* studioB, hash\* studioC, char\* name) {

    int opt;

    char\* film;

    printf("Berikut adalah daftar Film yang tersedia: \n");

    opt = getFilm(opt)-1;

    if (isFilmAvailable(filmList[opt])) {

        film = filmList[opt];

        if (opt == 0) opt = addCust(studioA, name, film);

        if (opt == 1) opt = addCust(studioB, name, film);

        if (opt == 2) opt = addCust(studioC, name, film);

        if (!opt) {

            if (filmSuggestion(film)) return buyTicket(studioA, studioB, studioC, name);

            else return NULL;

        }

    } else {

        if (filmSuggestion(filmList[opt])) return buyTicket(studioA, studioB, studioC, name);

        else return NULL;

    }

    return film;

}

int addCust(hash\* cinema, char\* name, char\* film) {

    hash custData = malloc(sizeof(struct hash\_t));

    custData->name = calloc(255, sizeof(char));

    custData->film = calloc(255, sizeof(char));

    custData->next = NULL;

    int hashVal = generateHash(name);

    strcpy(custData->name, name);

    strcpy(custData->film, film);

    if (cinema[hashVal] != NULL) {

        return 0;

    } else {

        cinema[hashVal] = custData;

    }

}

void getCinemaStats(hash\* obj) {

    for (int i=0; i<MAX\_ARR; i++) {

        hash temp = obj[i];

        if (temp != NULL) {

            printf("[%d] %s -> %s\n", i, temp->name, temp->film);

        }

    }

}

int getFilm() {

    int opt;

    for (int i=0; i<MAX\_NODE; i++) {

        printf("[%d] %s\n", i+1, filmList[i]);

    }

    printf("Masukkan pilihan anda: ");

    scanf("%d", &opt);

    return opt;

}

*static* char\* generateAlias(char\* name) {

    char\* temp = calloc(255, sizeof(char));

    char\* alias = calloc(100, sizeof(char));

    strcpy(temp, name);

    temp = strtok(temp, " ");

    strcat(alias, temp);

    do {

        temp = strtok(NULL, " ");

        if (temp) {

            char x = temp[0];

            strncat(alias, &x, 1);

        }

    } while (temp != NULL);

    free(temp);

    return alias;

}

*static* int generateHash(char\* name) {

    int val = 0;

    unsigned long ctr = 0;

    for (int i=0; i<strlen(name)-1; i++) {

        ctr += i+1 < 10 ? (int) name[i]\*10 + 1+1 : (int)name[i]\*100 + i+1;

    }

    ctr = (unsigned long)pow(ctr, 2);

    while(length(ctr) > 2) {

        val += ctr%10;

        ctr = ctr/10;

    }

    return val;

}

*static* unsigned length(*const* unsigned long num) {

    if (num < 10) return 1;

    return 1 + length(num/10);

}

int filmSuggestion(char\* film) {

    char ans[3];

    printf("%s belum ditayangkan saat ini, apakah anda ingin memesan film lainnya ?\n", film);

    printf("Jawaban [y/n]: ");

    scanf("%s", ans);

    if (ans[0] == 'y') {

        return 1;

    } else {

        return 0;

    }

}

*static* int isFilmAvailable(char\* film) {

    if (airingList[0] == 0) return 0;

    for (int i=0; i<3; i++) {

        if (!strcmp(film, airingList[i])) return i+1;

    }

    return 0;

}

*static* void parseTitle(char\* filename, char\* obj[6]) {

    FILE\* rawFile = fopen(filename, "r+");

    char\* buff = calloc(255, sizeof(char));

    char\* temp = calloc(255, sizeof(char));

    int ctr = -1;

    int isData = 0;

    while (fgets(buff, 255, rawFile)) {

        temp = strtok(buff, "\n");

        if (!strcmp(temp, "title\_Start")) isData = 1;

        if (!strcmp(temp, "title\_End")) isData = 0;

        if (isData && strcmp(buff, "title\_Start") && strcmp(buff, "title\_End")) {

            obj[++ctr] = strdup(temp);

        }

    }

}

*static* void parseFilm(char\* filename, int obj[MAX\_NODE][MAX\_NODE]) {

    FILE\* rawFile = fopen(filename, "r+");

    char\* buff = calloc(255, sizeof(char));

    char\* temp = calloc(255, sizeof(char));

    int isData = 0;

    int idx = -1;

    for (int i=0; i<MAX\_NODE; i++) {

        for (int j=0; j<MAX\_NODE; j++) {

            obj[i][j] = 0;

        }

    }

    while (fgets(buff, 255, rawFile)) {

        temp = strtok(buff, "\n");

        if (!strcmp(temp, "data\_start")) isData = 1;

        if (!strcmp(temp, "data\_end")) isData = 0;

        if (isData && strcmp(buff, "data\_start") && strcmp(buff, "data\_end")) {

            idx = atoi(strtok(temp, " "));

            do {

                temp = strtok(NULL, " ");

                obj[idx][atoi(temp)] = 1;

            } while (temp != NULL);

        }

    }

}

*static* void colorized(int matrices[MAX\_NODE][MAX\_NODE], int\* res) {

    int color[MAX\_NODE];

    color[0] = 0;

    int colorUsed[MAX\_NODE];

    for (int i=1; i<MAX\_NODE; i++) {

        color[i] = -1;

    }

    for (int i=0; i<MAX\_NODE; i++) {

        colorUsed[i] = 0;

    }

    for (int u=1; u<MAX\_NODE; u++) {

        for (int v=0; v<MAX\_NODE; v++) {

            if (matrices[u][v]) {

                if (color[v] != -1) {

                    colorUsed[color[v]] = 1;

                }

            }

        }

        int col;

        for (col=0; col<MAX\_NODE; col++) {

            if (!colorUsed[col]) break;

        }

        color[u] = col;

        for (int v=0; v<MAX\_NODE; v++) {

            if (matrices[u][v]) {

                if (color[v] != -1) {

                    colorUsed[color[v]] = 0;

                }

            }

        }

    }

    for (int x=0; x<MAX\_NODE; x++) res[x] = color[x];

}

*static* void orderCanceled() {

    printf("Mohon Maaf atas ketidaknyamanannya...\n");

}

**Modular: custServ.c**

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#include <conio.h>

#include "custServ.h"

void getPayment(char\* name, char\* film) {

    int money;

    int changes;

    char\* eq = calloc(255, sizeof(char));

    printf("Nama: %s\nFilm: %s\nBiaya: %d\n", name, film, TIC\_PRICE);

    printf("Masukkan nominal pembayaran: Rp. ");

    scanf("%d", &money);

    if (money < TIC\_PRICE) return;

    snprintf(eq, 255, "%d %d -", money, TIC\_PRICE);

    changes = calculate(eq);

    if (changes != 0) {

        printf("Kembalian anda: %d\nTerima kasih! :)\n\n", changes);

    }

};

void enter\_q(queue\* cust, char\* cusin) {

    queue newCust = malloc(sizeof(struct queue\_t));

    if (newCust == NULL) {

        printf("ERROR Occured!\n");

        return;

    }

    newCust->iden = strdup(cusin);

    newCust->link = NULL;

    if (\*cust == NULL) {

        \*cust = newCust;

        if (cust == NULL) {

            printf("E!\n");

            exit(-1);

        }

    } else {

        queue temp = \*cust;

        while (temp->link) {

            temp = temp->link;

        }

        temp->link = newCust;

    }

}

void display\_q(queue cust) {

    if (cust == NULL) return;

    queue temp = cust;

    printf("Luminous Cinema's Queue:\n");

    do {

        printf("[ ] %s\n", temp->iden);

        temp = temp->link;

    } while (temp != NULL);

}

char\* getCust(queue\* cust) {

    if (\*cust == NULL) return NULL;

    char\* name = calloc(255, sizeof(char));

    name = strdup((\*cust)->iden);

    if (!(\*cust)->link) {

        \*cust = NULL;

        return name;

    } else {

        \*cust = (\*cust)->link;

        return name;

    }

}

int getQueueSize(queue cust) {

    if (!cust) return 0;

    queue temp = cust;

    int ctr = 1;

    while (temp->link) {

        ctr++;

        temp = temp->link;

    }

    return ctr;

}

*static* void initStack(stack\* eqStack, size\_t size) {

    \*eqStack = malloc(sizeof(stack));

    (\*eqStack)->top = -1;

    (\*eqStack)->data = malloc(size\*sizeof(int));

}

*static* int isEmpty(stack eqStack) {

    return eqStack->top == -1;

}

*static* int peek(stack eqStack) {

    return eqStack->data[eqStack->top];

}

*static* void push(stack \*eqStack, int val) {

    (\*eqStack)->data[++(\*eqStack)->top] = val;

}

*static* int pop(stack \*eqStack) {

    if (!isEmpty(\*eqStack)) {

        return (\*eqStack)->data[(\*eqStack)->top--];

    }

    return -1;

}

*static* int isOperand(char val) {

    return ((val - '0') >= 0 && (val - '0') <= 9);

}

*static* int calculate(char\* exp) {

    stack\* eqStack;

    initStack(eqStack, strlen(exp));

    if (!eqStack) return -1;

    for (int i=0; exp[i]; i++) {

        if (exp[i] == ' ') continue;

        else if (isOperand(exp[i])) {

            int num = 0;

            while (isOperand(exp[i])) {

                num = num\*10 + (int)(exp[i]-'0');

                i++;

            }

            i--;

            push(eqStack, num);

        } else {

            int val1 = pop(eqStack);

            int val2 = pop(eqStack);

            switch (exp[i]) {

                case '+': push(eqStack, val2+val1); break;

                case '-': push(eqStack, val2-val1); break;

                case '\*': push(eqStack, val2\*val1); break;

                case '/': push(eqStack, val2/val1); break;

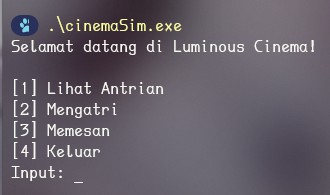
            }

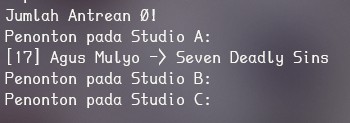
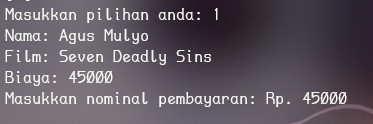
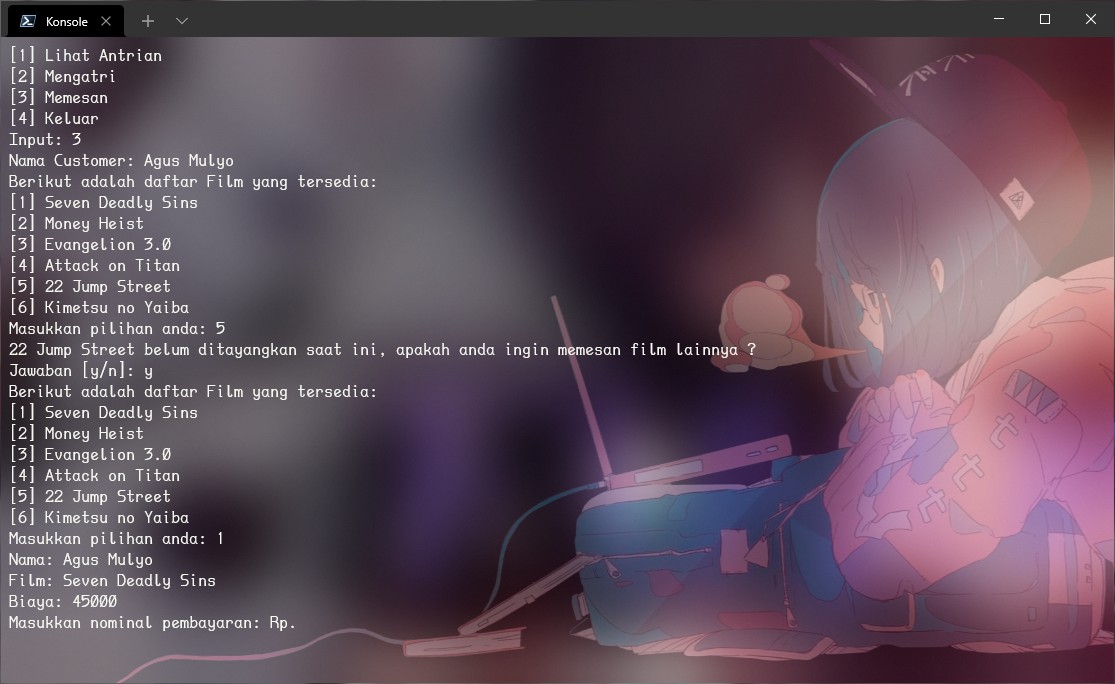
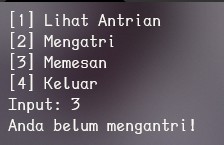
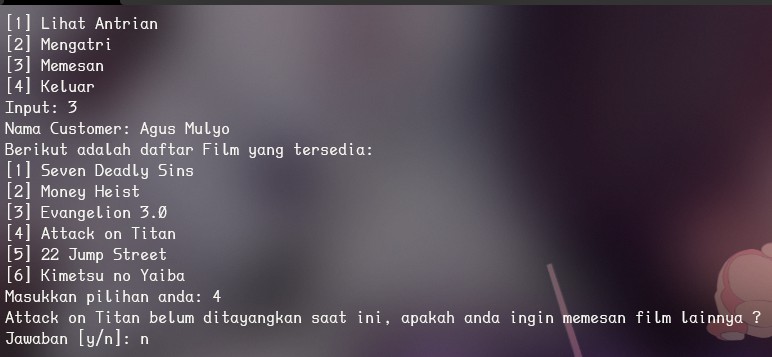
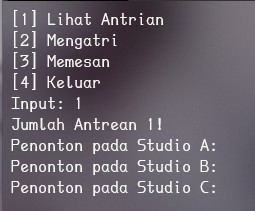
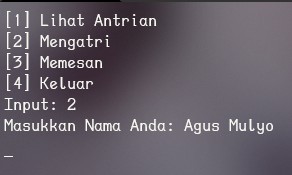
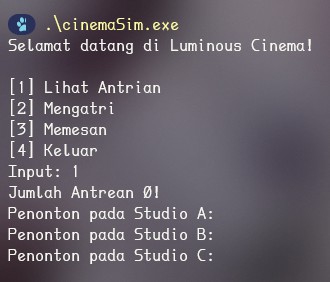
        }

    }

    return pop(eqStack);

}

* 1. **Output**

****

1. **Kesimpulan**

Berdasarkan hasil tugas akhir ini dapat disimpulkan bahwa sistem pembelian tiket bioskop menggunakan bahasa c cocok digunakan untuk memepermudah pembeli membeli tiket secara langsung maupun manual.