Movie Database Application.

2019년 1학기 자료구조 설계 중간보고서

2017103733 소프트웨어융합학과 우수진

What is Movie Database Application?

Why?

현대 사회에서 영화는 가장 대중적인 문화생활 중 하나로 자리잡고 있고 있다. 이에 현대인들이 SNS, 사진, 블로그 등을 통해 자신의 일상의 기록하는 모습이 떠올랐다. 이제 영화도 우리 일상의 한 부분이 되어가고 있는 만큼, 이제는 이를 기록할 수 있는 서비스도 필요하다고 생각한다. 따라서 'Movie Database Application'을 개발하게 되었다.

Provided Functions

1. Add Movie
2. Delete Movie
3. Add cast
4. Delete cast
5. Find Movie
6. Find Movie Actor
7. Find Movie Rating
8. Find Movie Year
9. Find Movie Period
10. Print Movies
11. Exit

Welcome to the movie information database program.

Interface

```
Welcome to the movie information database program.
********************************
Available Operations:
        1.  Add movie
        2.  Delete movie
        3.  Add cast name to movie
        4.  Delete cast name from movie
        5.  Find movie
        6.  Find movies by actor name
        7.  Find movies by rating
        8.  Find movies by year
        9.  Find movies by time period
        10. Print movies in order
        11. Exit
********************************
Enter your option number:
```

Provided Functions

1. Add Movie
2. Delete Movie
3. Add cast
4. Delete cast
5. Find Movie
6. Find Movie Actor
7. Find Movie Rating
8. Find Movie Year
9. Find Movie Period
10. Print Movies
11. Exit

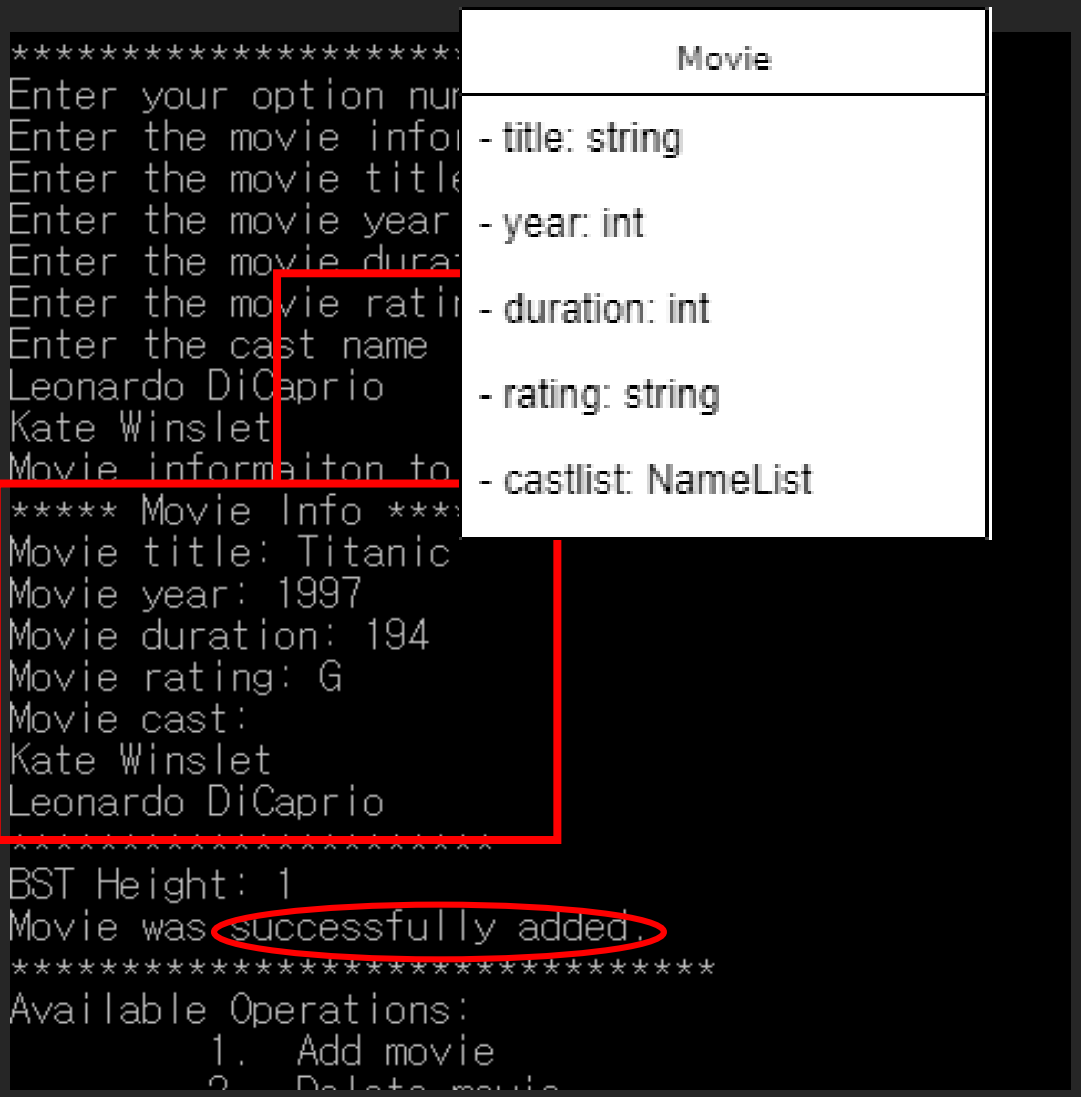
Ex) 1. Add Movie

```
**********************************
Enter your option number: 1
Enter the movie information
Enter the movie title: Titanic
Enter the movie year: 1997
Enter the movie duration: 194
Enter the movie rating: G
Enter the cast name list: 2
Leonardo DiCaprio
Kate Winslet
Movie informaiton to be added:
***** Movie Info *****
Movie title: Titanic
Movie year: 1997
Movie duration: 194
Movie rating: G
Movie cast:
Kate Winslet
Leonardo DiCaprio
**********************
BST Height: 1
Movie was successfully added.
**********************************
Available Operations:
        1.  Add movie
```

Provided Functions

1. Add Movie
2. Delete Movie
3. Add cast
4. Delete cast
5. Find Movie
6. Find Movie Actor
7. Find Movie Rating
8. Find Movie Year
9. Find Movie Period
10. Print Movies
11. Exit

Ex) 1. Add Movie



Provided Functions

1. Add Movie
2. Delete Movie
3. Add cast
4. Delete cast
5. Find Movie
6. Find Movie Actor
7. Find Movie Rating
8. Find Movie Year
9. Find Movie Period
10. Print Movies
11. Exit

Ex) 1. Add Movie

```
**********************************
Enter your option number: 1
Enter the movie information
Enter the movie title: Titanic
Enter the movie year: 1997
Enter the movie duration: 194
Enter the movie rating: G
Enter the cast name list: 2
Leonardo DiCaprio
Kate Winslet
Movie informaiton to be added:
***** Movie Info *****
Movie title: Titanic
Movie year: 1997
Movie duration: 194
Movie rating: G
Movie cast:
Kate Winslet
Leonardo DiCaprio
**********************
BST Height: 1
Movie was successfully added.
**********************************
Available Operations:
        1.  Add movie
```

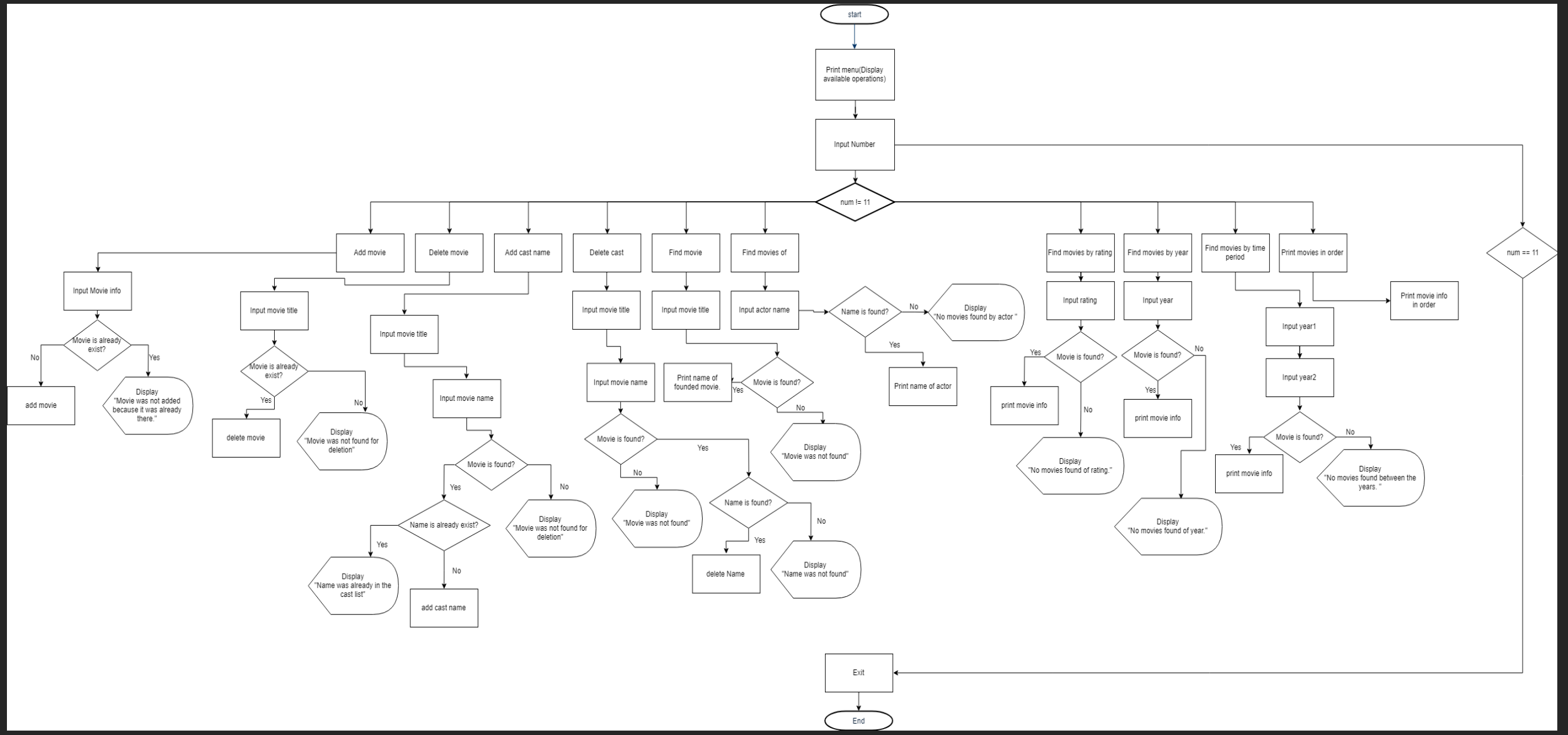
Before you Enter the input number which is 11,

the application is not Exterminated.

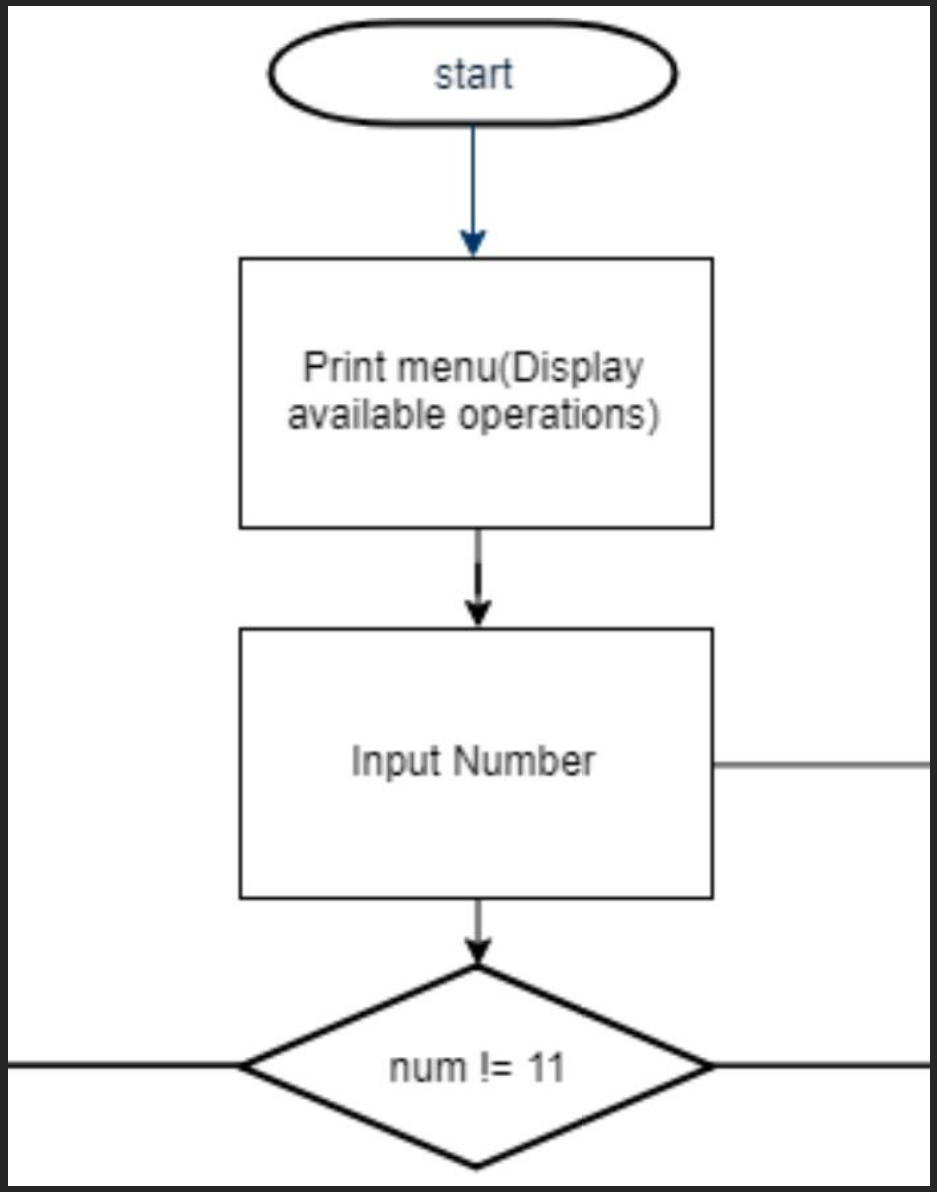
```
int get_option() {
    int num = 0;
    cout << "Enter your option number: ";
    cin.clear();
    cin >> num;

    if (num == 11) {
        return 0;
    }
    else {
        return num;
    }
}
```

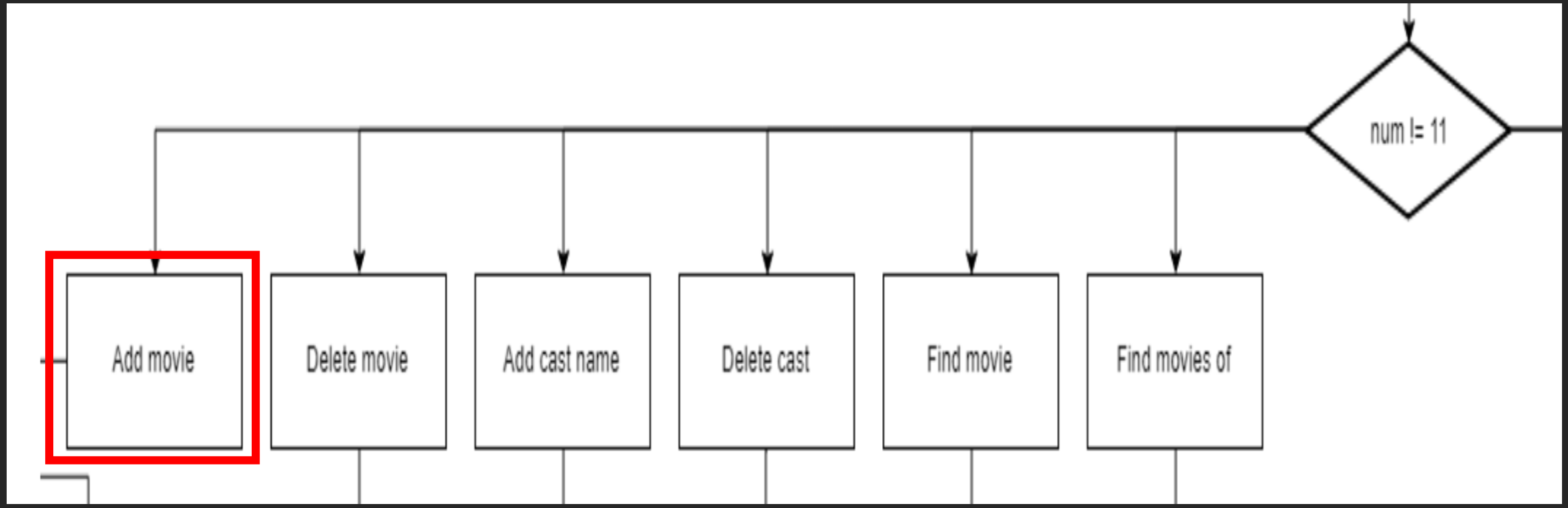
Flow chart


start
Print menu(Display available operations)
Input Number
num != 11
num == 11
Add movie
Delete movie
Add cast name
Delete cast
Find movie
Find movies of
Find movies by rating
Find movies by year
Find movies by time period
Print movies in order
Input Movie info
Movie is already exist?
No
Yes
add movie
Display "Movie was not added because it was already there."
Input movie title
Movie is already exist?
Yes
No
delete movie
Display "Movie was not found for deletion"
Input movie title
Input movie name
Movie is found?
Yes
No
Name is already exist?
Yes
No
Display "Name was already in the cast list"
add cast name
Display "Movie was not found for deletion"
Input movie title
Input movie name
Movie is found?
Yes
No
Display "Movie was not found"
Name is found?
Yes
No
delete Name
Display "Name was not found"
Input movie title
Print name of founded movie.
Movie is found?
Yes
No
Display "Movie was not found"
Input actor name
Name is found?
No
Display "No movies found by actor "
Yes
Print name of actor
Input rating
Movie is found?
Yes
print movie info
No
Display "No movies found of rating."
Input year
Movie is found?
No
Yes
print movie info
Display "No movies found of year."
Input year1
Input year2
Movie is found?
Yes
print movie info
No
Display "No movies found between the years. "
Print movie info in order
Exit
End

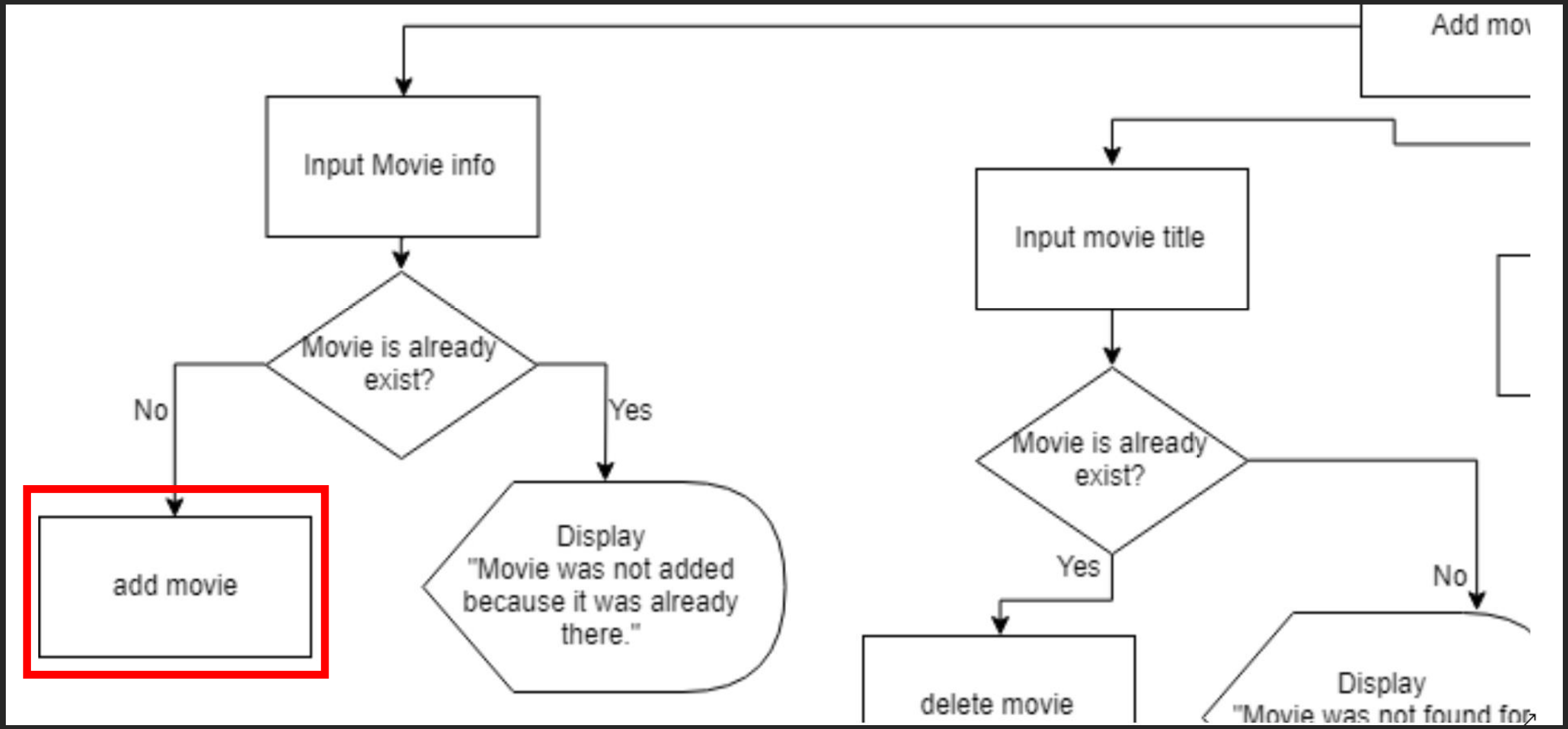
Ex) Input num == 1



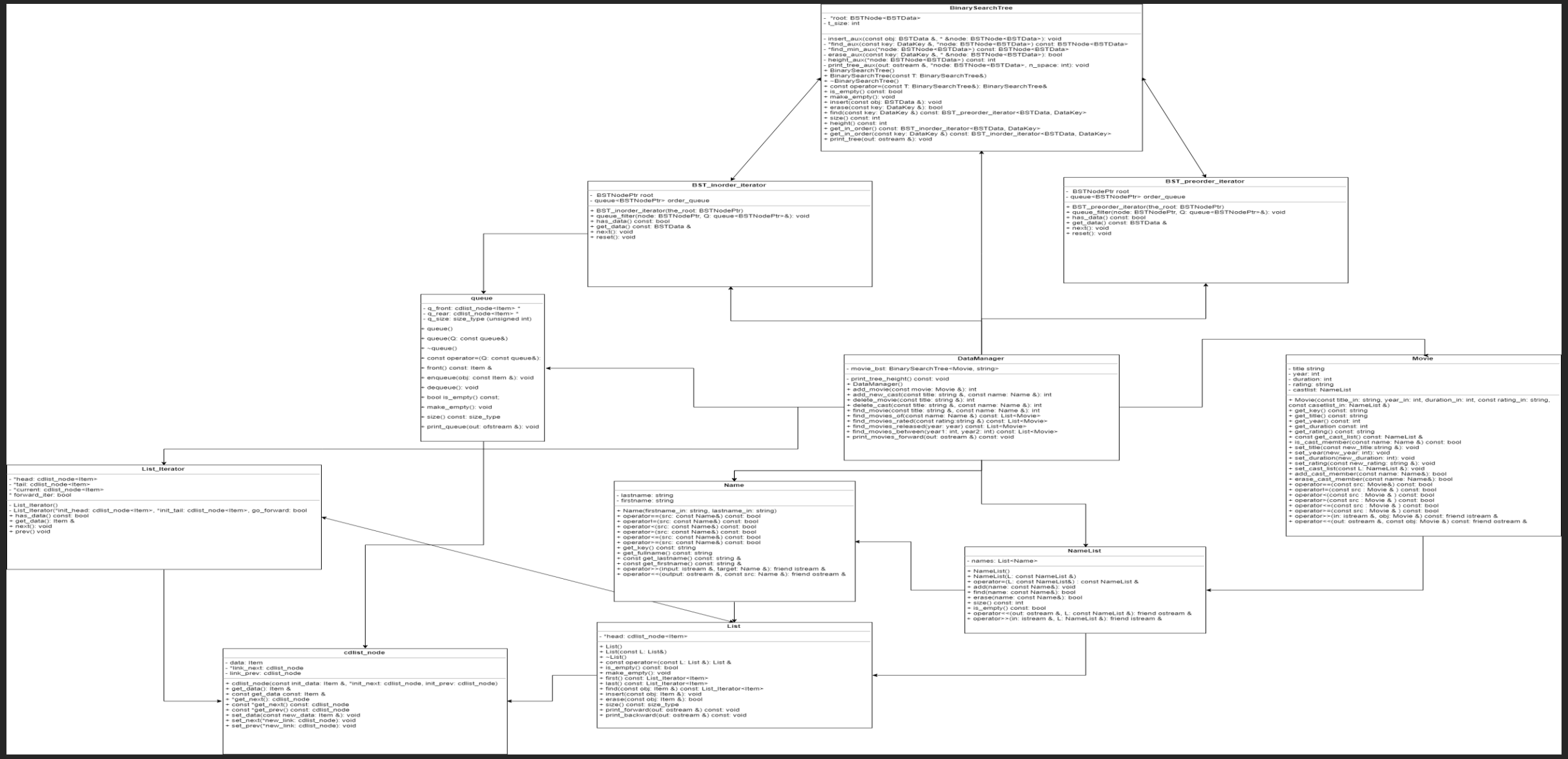
Ex) Input num == 1



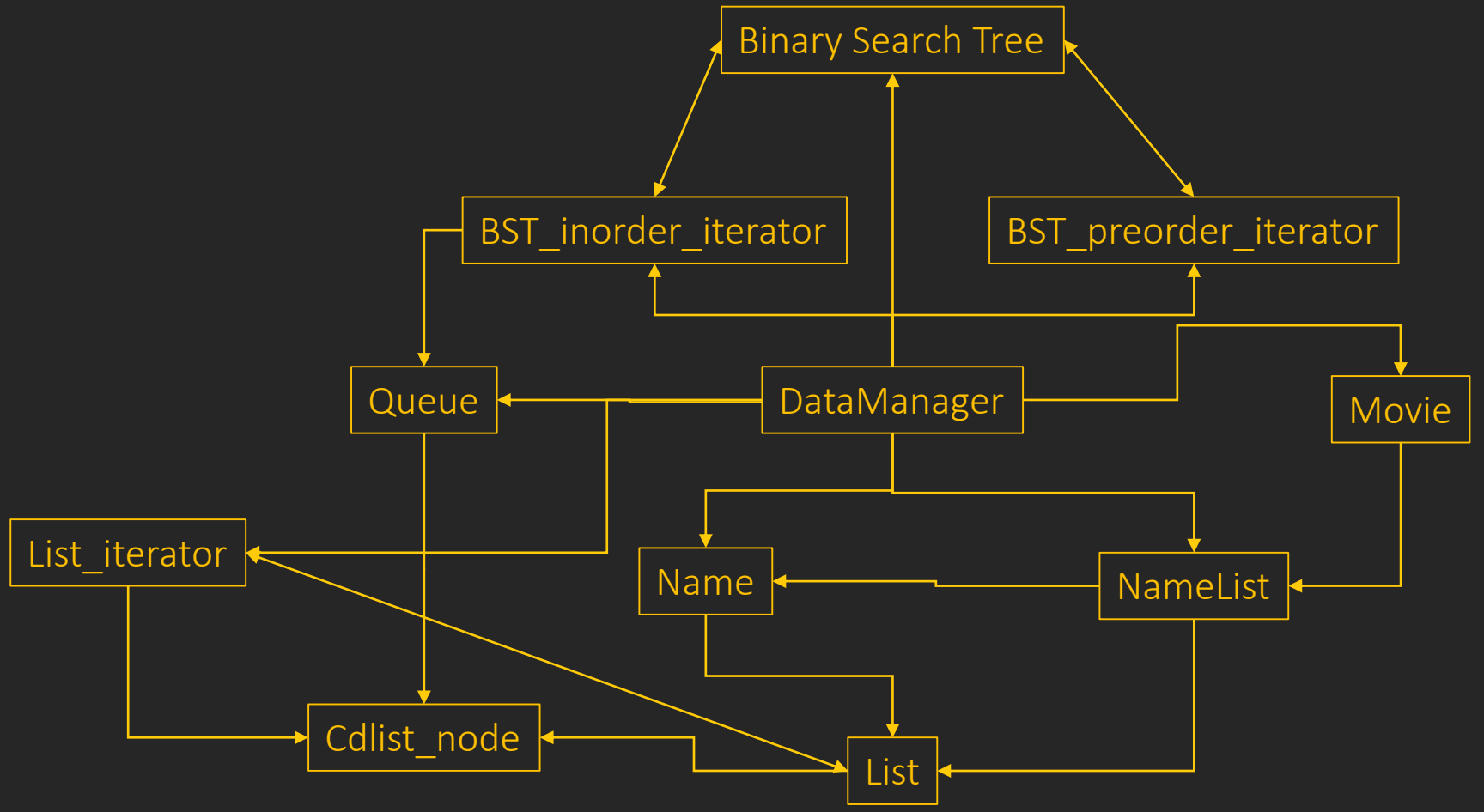
Ex) Input num == 1



Class diagram



Class Diagram (Simple ver.)



Data manager class

DataManager
- movie_bst: BinarySearchTree<Movie, string>
- print_tree_height() const: void + DataManager() + add_movie(const movie: Movie &): int + add_new_cast(const title: string &, const name: Name &): int + delete_movie(const title: string &): int + delete_cast(const title: string &, const name: Name &): int + find_movie(const title: string &, const name: Name &): int + find_movies_of(const name: Name &) const: List<Movie> + find_movies_rated(const rating:string &) const: List<Movie> + find_movies_released(year: year) const: List<Movie> + find_movies_between(year1: int, year2: int) const: List<Movie> + print_movies_forward(out: ostream &) const: void

Mainly Used Data structure

- Queue
- Sorted Circular Doubly Linked List
- Binary Search Tree

Doubly Linked List

Q. Array? Linked List?

A. Linked List, Array는 index를 알 때 데이터 접근 속도가 빠르다. 그러나 삽입 및 삭제의 경우 Linked List가 훨씬 용이하다.

Q. Linked List? Doubly Linked List?

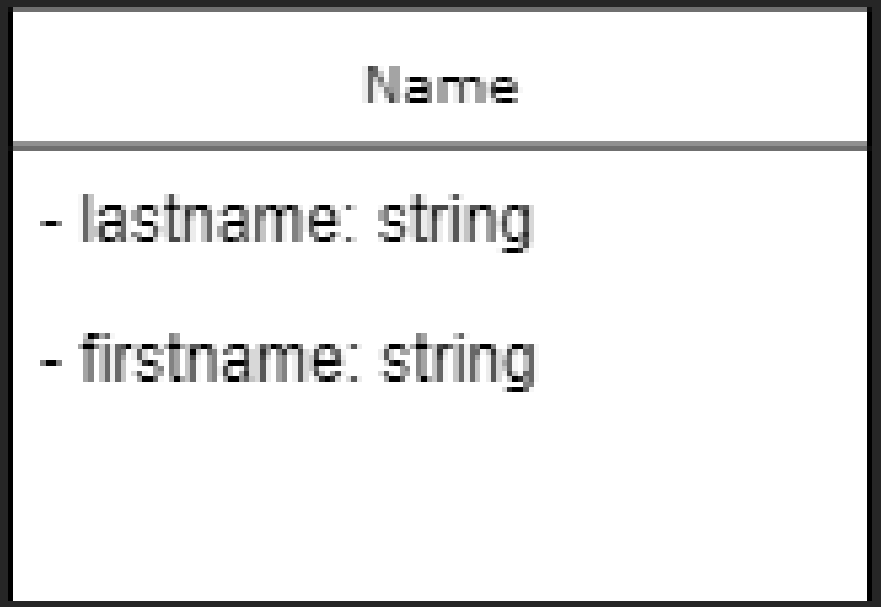
A. Doubly Linked List, Linked List는 다음 노드에 대한 탐색만 가능하지만, Doubly Linked List는 두 개의 포인터가 있어 앞뒤로 탐색이 가능하다. 따라서 일반적인 Linked List에 비해 효율적인 탐색이 가능하다. (최대 탐색 시간 반으로 단축)

Binary Search Tree

Q. Binary Search Tree? List?

A. Binary Search Tree, Binary Search Tree는 Binary Search(이진 탐색)와 Linked List(연결 리스트)를 결합한 자료구조의 일종이다. Binary Search Tree의 경우 균형이 잡혀있는 한, 검색 경로가 Linked List 보다 훨씬 짧다.(LogN)

그러나 Name 정보를 다룰 때에는
검색보다 삽입 및 삭제가 더 빈번하게 발생하므로
List를 이용하는 것이 더 적절하다고 판단하였다.



Add Cast

Description of used data structure

4. add()

If(Movie title은 존재o && name은 존재x)
-> sorted order가 되도록 적절한 위치에 Insert

3. Find()

- Sorted Circular Doubly Linked List에 name이 존재하는지 탐색

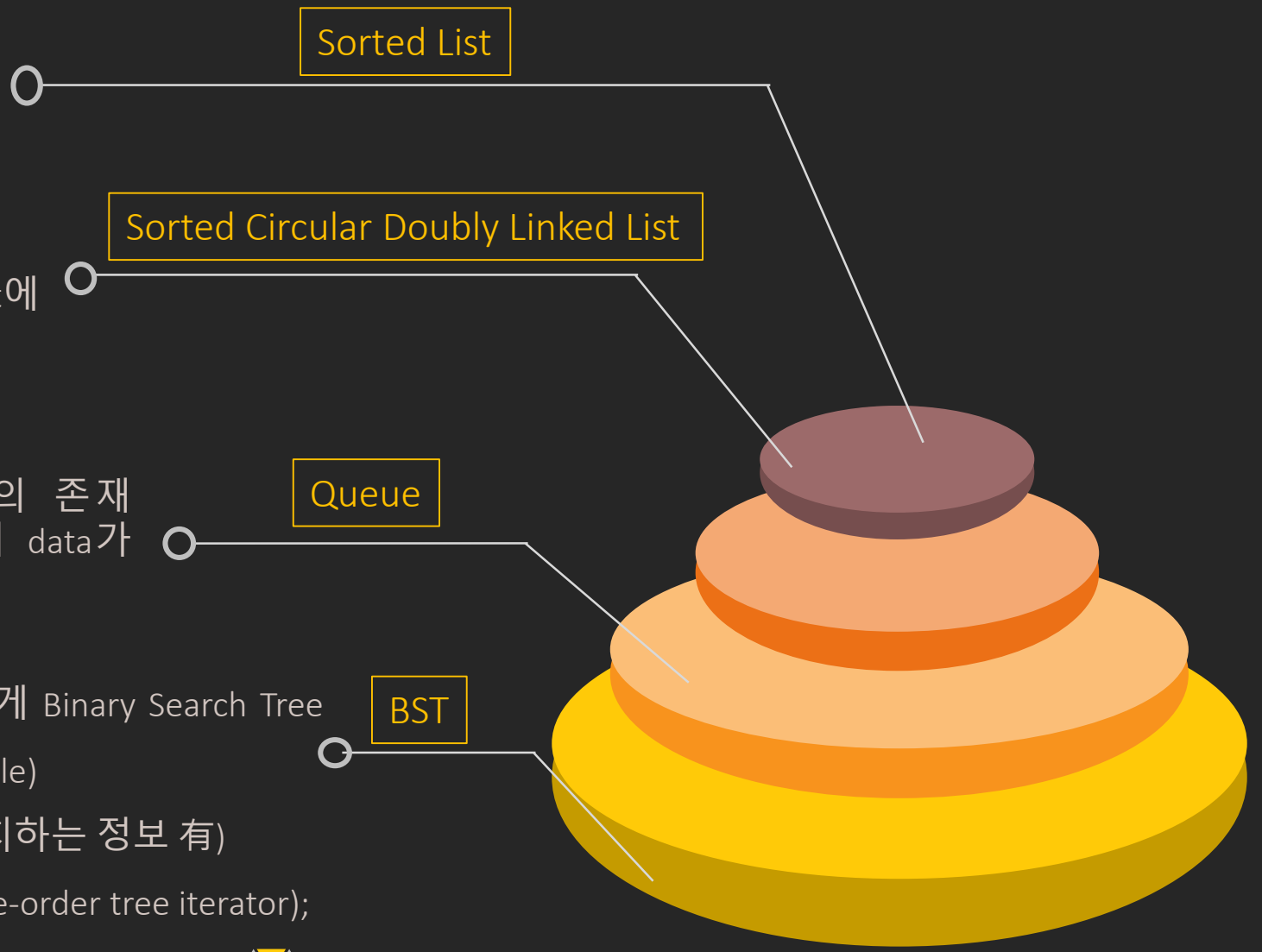
2. has_data()

- Iterator를 통해 Movie의 존재 여부 판단 (= iterator에 data가 있는가)

1. Find()

- Recursive하게 Binary Search Tree 탐색(key: title)
- if(key와 일치하는 정보 有)

  Return (pre-order tree iterator);



Add Cast

Description of used data structure

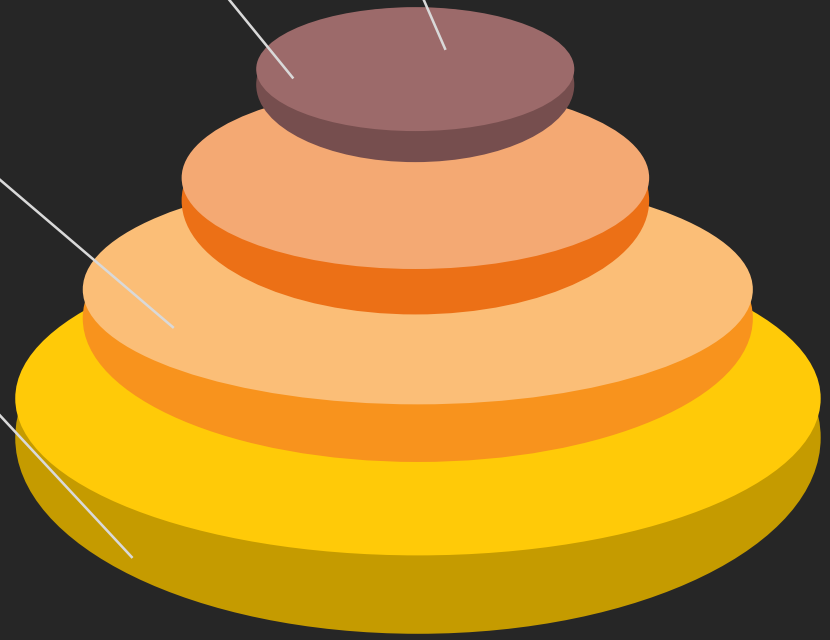
Sorted List

Sorted Circular Doubly Linked List

Queue

BST

Name
- lastname: string
- firstname: string



4. add()

If(Movie title은 존재o && name은 존재x)
-> sorted order가 되도록 적절한 위치에 Insert

3. Find()

- Sorted Circular Doubly Linked List에 name이 존재하는지 탐색

2. has_data()

- Iterator를 통해 Movie의 존재 여부 판단 (= iterator에 data가 있는가)

1. Find()

- Recursive하게 Binary Search Tree 탐색(key: title)
- if(key와 일치하는 정보 有)

  Return (pre-order tree iterator);

Soojin Woo

Add Movie

Description of used data structure

Insert()

If(movie가 존재하지 x)

-> Binary Search Tree.recursive_insert(movie info);

Binary Search Tree

Has_data()

- Iterator를 통해 Movie의 존재 여부 판단 (=iterator에 data가 있는가)

Queue

Find()

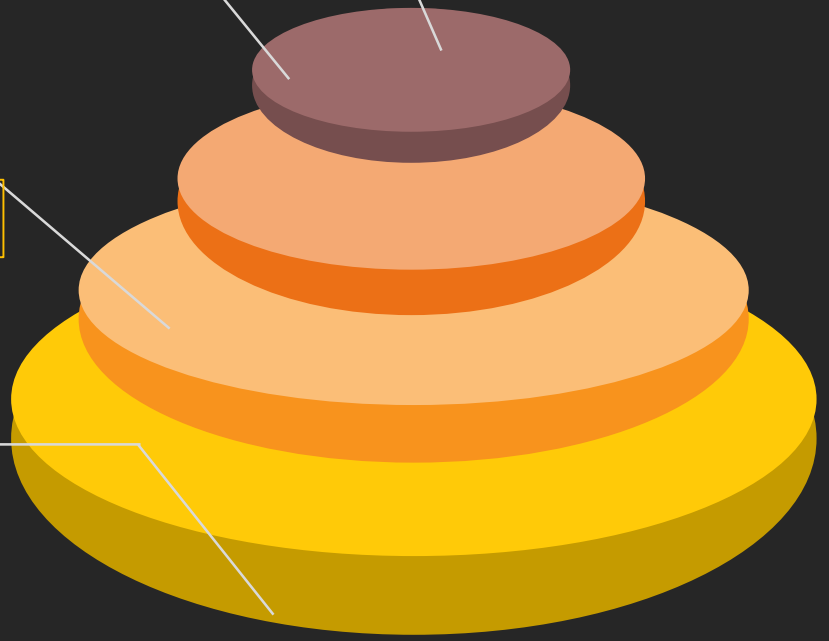
- Recursive하게 Binary Search Tree 탐색
- if(key와 일치하는 정보 有)

  -> Return (pre-order tree iterator);

Binary Search Tree

Get_key()

Movie에 대한 key를 반환 (key: title)



Thank You

2019년 1학기 자료구조 설계 중간보고서

2017103733 소프트웨어융합학과 우수진