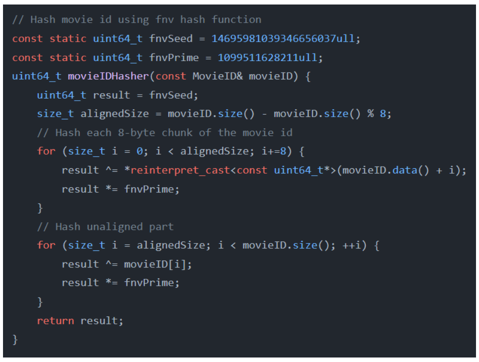
Movies Database: Team 6

Ulysses Morgan
Abdullah Malik
Arnesh Regmi
Sunho Kim

Application Data

- Primary Key (String)
- Title (String)
- Language (String)
- Director (String)
- Release date (Int)
- Movie Rating (Bool)

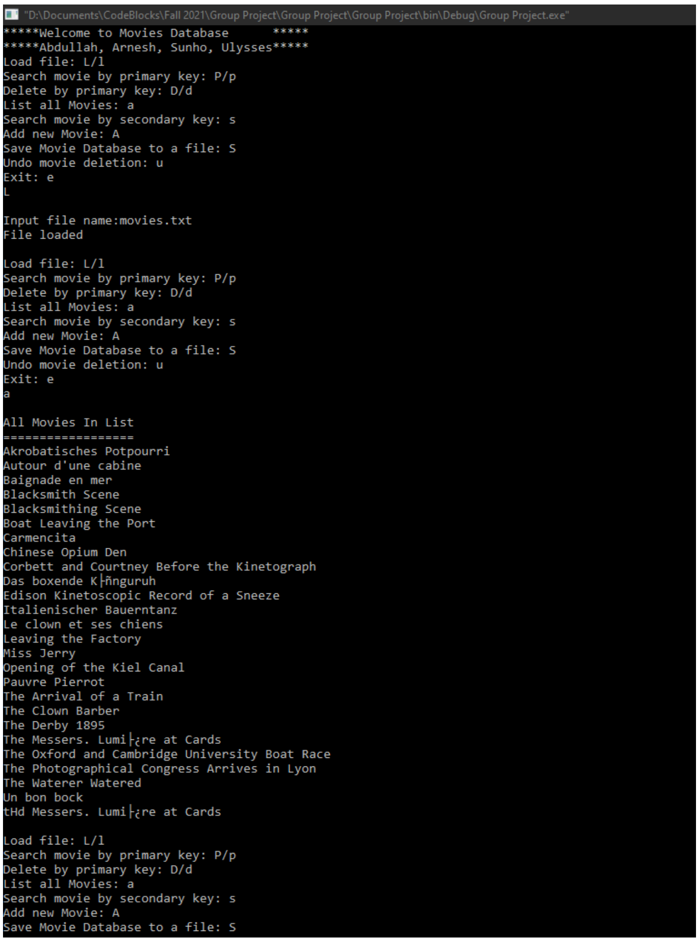
Hash Function



```
// Hash movie id using fnv hash function
const static uint64_t fnvSeed = 14695981039346656037ull;
const static uint64_t fnvPrime = 1099511628211ull;
uint64_t movieIDHasher(const MovieID& movieID) {
    uint64_t result = fnvSeed;
    size_t alignedSize = movieID.size() - movieID.size() % 8;
    // Hash each 8-byte chunk of the movie id
    for (size_t i = 0; i < alignedSize; i+=8) {
        result ^= *reinterpret_cast<const uint64_t*>(movieID.data() + i);
        result *= fnvPrime;
    }
    // Hash unaligned part
    for (size_t i = alignedSize; i < movieID.size(); ++i) {
        result ^= movieID[i];
        result *= fnvPrime;
    }
    return result;
}
```

Our hash function (Fowler–Noll–Vo) was the default hash function used by python before, and it's known for its simplicity and fastness. It uses bitwise XOR operation to "mix" the hash, which is a really cheap computation and changes the bit pattern abruptly.

Sample Output

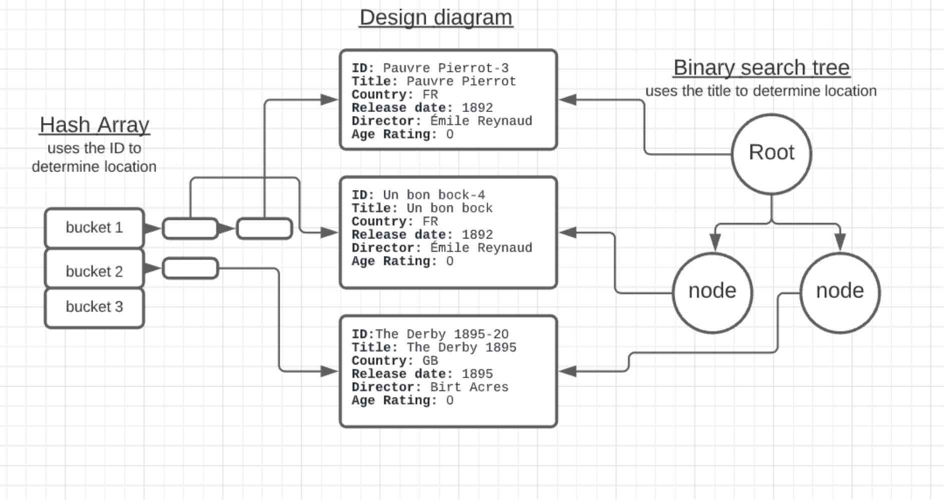


```
"D:\Documents\CodeBlocks\Fall 2021\Group Project\Group Project\Group Project\bin\Debug\Group Project.exe"
*****Welcome to Movies Database      *****
*****Abdullah, Arnesh, Sunho, Ulysses*****
Load file: L/l
Search movie by primary key: P/p
Delete by primary key: D/d
List all Movies: a
Search movie by secondary key: s
Add new Movie: A
Save Movie Database to a file: S
Undo movie deletion: u
Exit: e
L

Input file name:movies.txt
File loaded

Load file: L/l
Search movie by primary key: P/p
Delete by primary key: D/d
List all Movies: a
Search movie by secondary key: s
Add new Movie: A
Save Movie Database to a file: S
Undo movie deletion: u
Exit: e
a

All Movies In List
==================
Akrobatisches Potpourri
Autour d'une cabine
Baignade en mer
Blacksmith Scene
Blacksmithing Scene
Boat Leaving the Port
Carmencita
Chinese Opium Den
Corbett and Courtney Before the Kinetograph
Das boxende K├ñnguruh
Edison Kinetoscopic Record of a Sneeze
Italienischer Bauerntanz
Le clown et ses chiens
Leaving the Factory
Miss Jerry
Opening of the Kiel Canal
Pauvre Pierrot
The Arrival of a Train
The Clown Barber
The Derby 1895
The Messers. Lumi├¿re at Cards
The Oxford and Cambridge University Boat Race
The Photographical Congress Arrives in Lyon
The Waterer Watered
Un bon bock
tHd Messers. Lumi├¿re at Cards

Load file: L/l
Search movie by primary key: P/p
Delete by primary key: D/d
List all Movies: a
Search movie by secondary key: s
Add new Movie: A
Save Movie Database to a file: S
```

Comments

What our code does: We made a database dedicated to storing movies. It takes a file with movies as input. It holds various information about movies and allows for the user to insert and delete movies from the database. Finally, when the user is done with inserting/deleting movies the information is written down on a newly created file.

Working as a Team: The assignment proved to be a challenge but we managed to work together by keeping up with each other weekly. As a team of 4, we distributed the project by assigning different tasks. We made sure to keep a group chat and update everybody on any confusions they may have had.



Design diagram

Hash Array
uses the ID to determine location

Binary search tree
uses the title to determine location

ID: Pauvre Pierrot-3
Title: Pauvre Pierrot
Country: FR
Release date: 1892
Director: Émile Reynaud
Age Rating: 0

ID: Un bon bock-4
Title: Un bon bock
Country: FR
Release date: 1892
Director: Émile Reynaud
Age Rating: 0

ID: The Derby 1895-20
Title: The Derby 1895
Country: GB
Release date: 1895
Director: Birt Acres
Age Rating: 0