**a. Team**

Name: Recurssive inception

Member:

Braddon

Jie

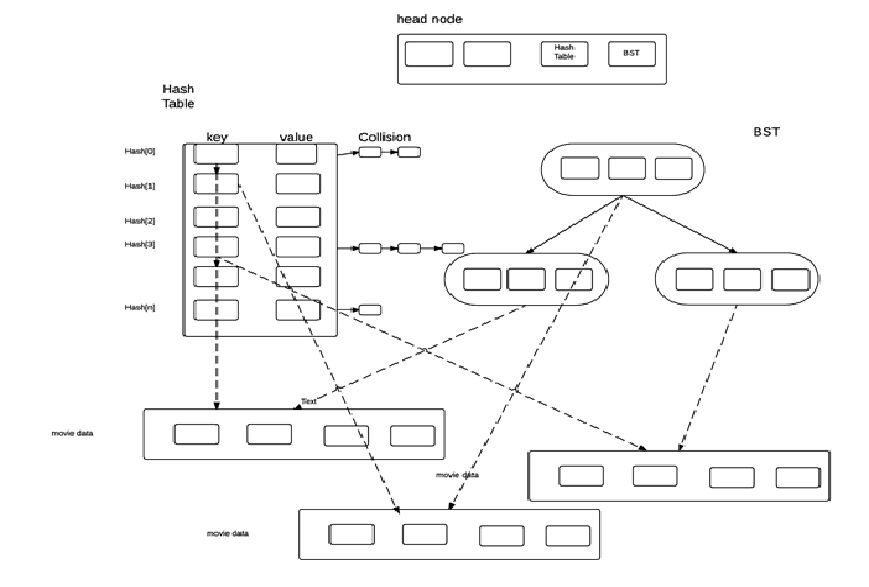
Xiaojing

David

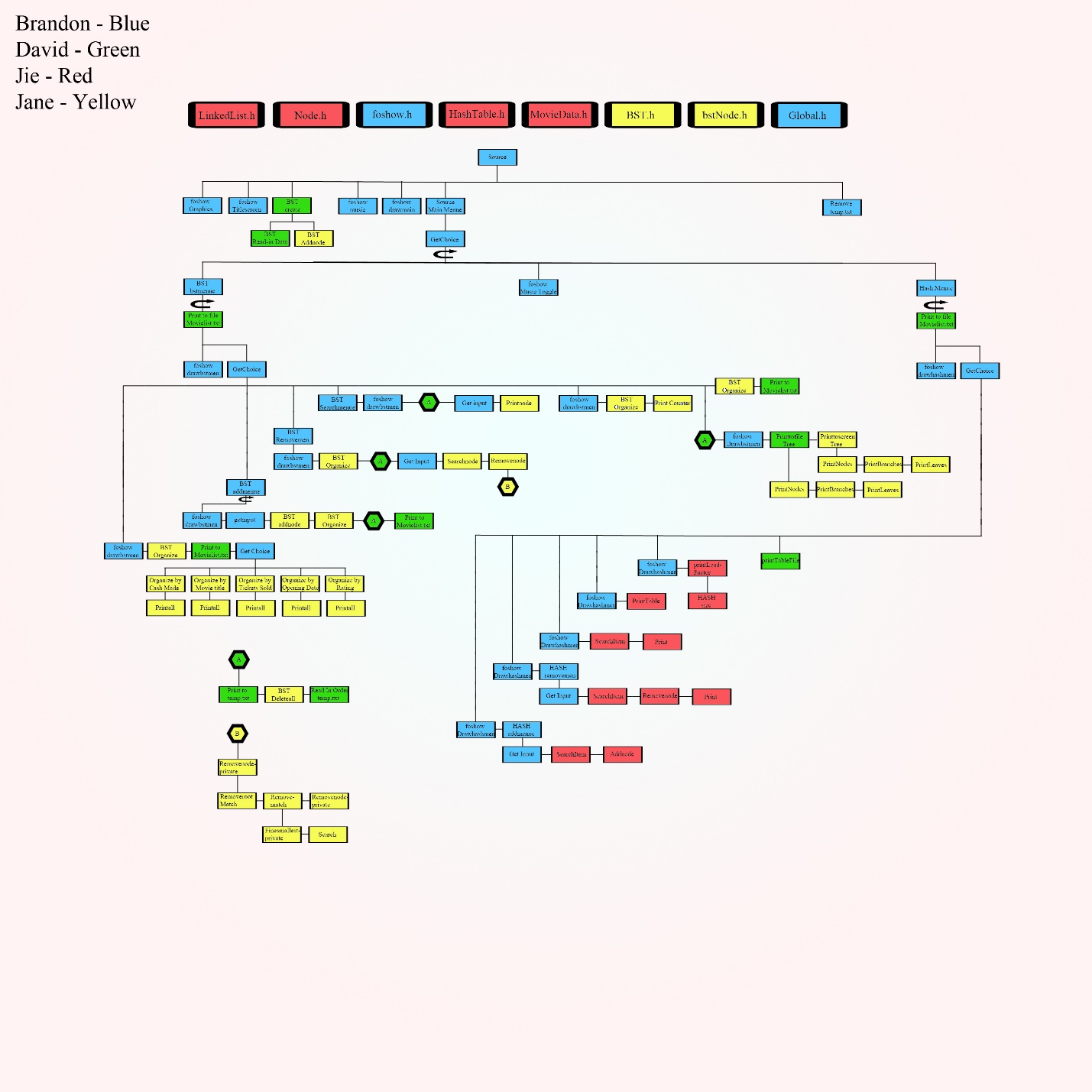
**b. Introduction**

This program uses BST and Hashtables to hold movie data.

**c. Data Diagram**

****

**d. Flowchart**

****

**e. Member’s task**

Braddon – team leader (code main and menu functions, etc)

Jie – hash tables (code hash functions include collision and print statics)

Xiaojing – BST (code bst functions)

David – Standard IO and file IO (code standard and file IO functions)

**f. Hash**

**Hash Function**

int HashTable::getHashIndex(string itemKey)

{

int index = 0;

for (int i = 0; i < itemKey.length(); i++)

{

index += (int)itemKey[i]; // sum of asc value of each char

}

index = index % myTable.size(); // Modulo Division

return index;

}

**Collision Resolution: Linked List**

Get hash index for by using the key of new item

If ( headptr -> null)

{

Insert the new item

}

Else

{

Insert new node before the existing item

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Index | Key | Value |  |  |  |  |  |
| 0 | Borne Identity | Borne Identity | X |  |  |  |  |
| 1 | Spectre | Spectre | X |  |  |  |  |
| 2 | Toy Story 3 | Toy Story 3 |  | → | Story Toy 3 | → | Iron Man 3 |
| 3 | Frozen | Frozen | X |  |  |  |  |
| 4 |  |  | X |  |  |  |  |
| 5 |  |  | X |  |  |  |  |
| 6 | Matrix | Matrix |  | → | Hancock |  |  |
| 7 |  |  | X |  |  |  |  |
|  | … |  |  |  |  |  |  |
|  | … |  |  |  |  |  |  |
|  | … |  |  |  |  |  |  |
| 24 |  |  | X |  |  |  |  |
| 25 | Secondhand Lions | Secondhand Lions | X |  |  |  |  |