Nirbhay Sharma (B19CSE114)
Lab -3

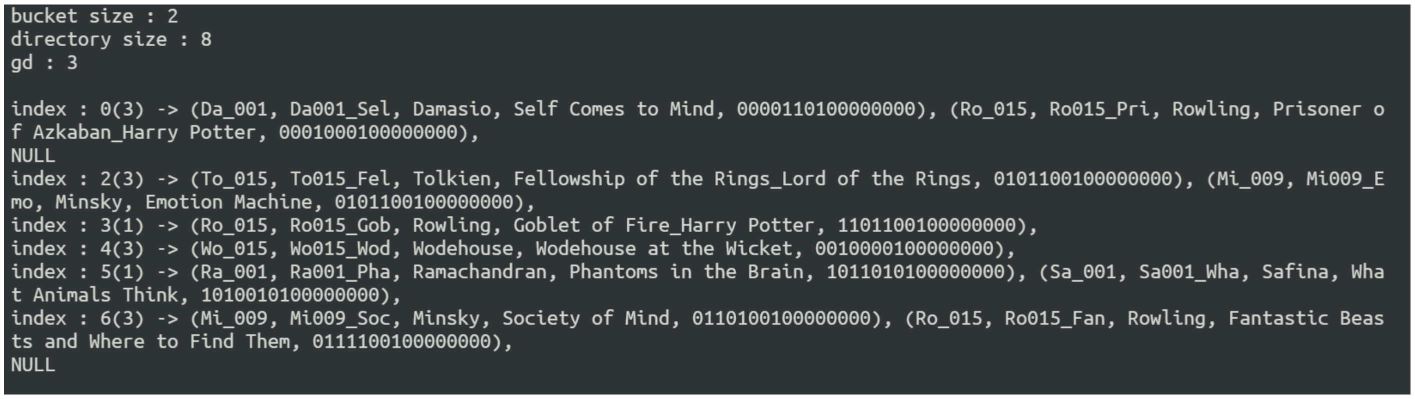
Ans 1)
In my opinion the potential candidate keys in this are book_name (considering copyright issues for the same name) and one can be book_id as it is very rare that the same author will write a book that has same initial 3 letters so for the time being this is also a potential candidate key

Primary key here is of course book_name as it can uniquely identify all the attributes in all the worst cases which is not the case with the book_id as primary key

Ans 2)
Extendible hashing code is provided in the given zip file which tries to optimize the search time of the books

Ans 3)
Yes larger bucket size would be more helpful in the fact that the overflows occurs somewhat late when we have large bucket size and still the searching time will be O(size of bucket) which is anyways constant as our size of bucket is constant

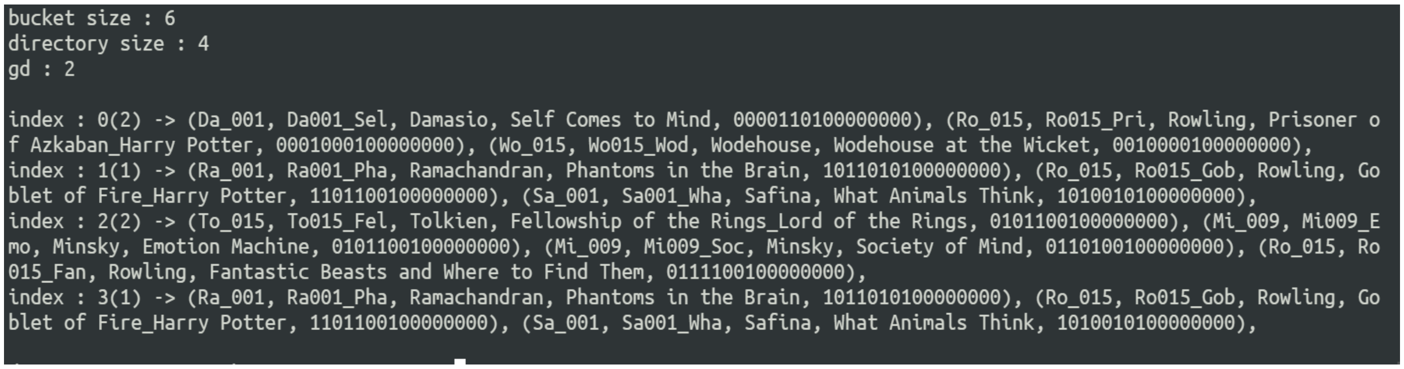
Experimental results:
For bucket size 2



```
bucket size : 2
directory size : 8
gd : 3

index : 0(3) -> (Da_001, Da001_Sel, Damasio, Self Comes to Mind, 0000110100000000), (Ro_015, Ro015_Pri, Rowling, Prisoner o
f Azkaban_Harry Potter, 0001000100000000),
NULL
index : 2(3) -> (To_015, To015_Fel, Tolkien, Fellowship of the Rings_Lord of the Rings, 0101100100000000), (Mi_009, Mi009_E
mo, Minsky, Emotion Machine, 0101100100000000),
index : 3(1) -> (Ro_015, Ro015_Gob, Rowling, Goblet of Fire_Harry Potter, 1101100100000000),
index : 4(3) -> (Wo_015, Wo015_Wod, Wodehouse, Wodehouse at the Wicket, 0010000100000000),
index : 5(1) -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Sa_001, Sa001_Wha, Safina, Wha
t Animals Think, 1010010100000000),
index : 6(3) -> (Mi_009, Mi009_Soc, Minsky, Society of Mind, 0110100100000000), (Ro_015, Ro015_Fan, Rowling, Fantastic Beas
ts and Where to Find Them, 0111100100000000),
NULL
```

Clearly visible here that since bucket size is 2 so at the final table we have our directory size as 8

For bucket size 6
Here also clearly visible that when our bucket size is 6 then we can accommodate more elements in one bucket so automatically the directory size is less



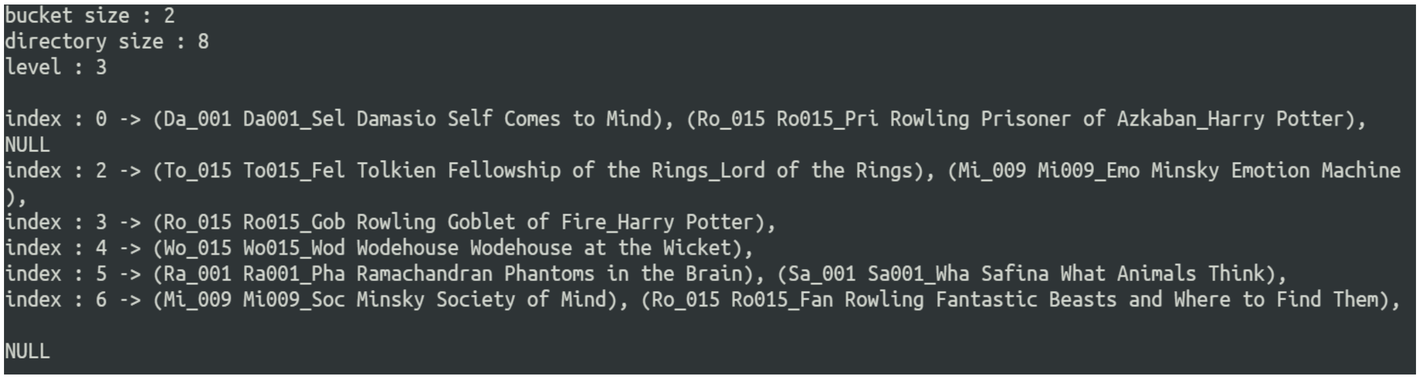
```
bucket size : 6
directory size : 4
gd : 2

index : 0(2) -> (Da_001, Da001_Sel, Damasio, Self Comes to Mind, 0000110100000000), (Ro_015, Ro015_Pri, Rowling, Prisoner o
f Azkaban_Harry Potter, 0001000100000000), (Wo_015, Wo015_Wod, Wodehouse, Wodehouse at the Wicket, 0010000100000000),
index : 1(1) -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Ro_015, Ro015_Gob, Rowling, Go
blet of Fire_Harry Potter, 1101100100000000), (Sa_001, Sa001_Wha, Safina, What Animals Think, 1010010100000000),
index : 2(2) -> (To_015, To015_Fel, Tolkien, Fellowship of the Rings_Lord of the Rings, 0101100100000000), (Mi_009, Mi009_E
mo, Minsky, Emotion Machine, 0101100100000000), (Mi_009, Mi009_Soc, Minsky, Society of Mind, 0110100100000000), (Ro_015, Ro
015_Fan, Rowling, Fantastic Beasts and Where to Find Them, 0111100100000000),
index : 3(1) -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Ro_015, Ro015_Gob, Rowling, Go
blet of Fire_Harry Potter, 1101100100000000), (Sa_001, Sa001_Wha, Safina, What Animals Think, 1010010100000000),
```

Ans 4)
linear hashing code is provided in the given zip file which tries to optimize the search time of the books

Ans 5)
Yes larger bucket size would be more helpful in the fact that the overflows occurs somewhat late when we have large bucket size and still the searching time will be O(size of bucket) which is anyways constant as our size of bucket is constant

Experimental results:

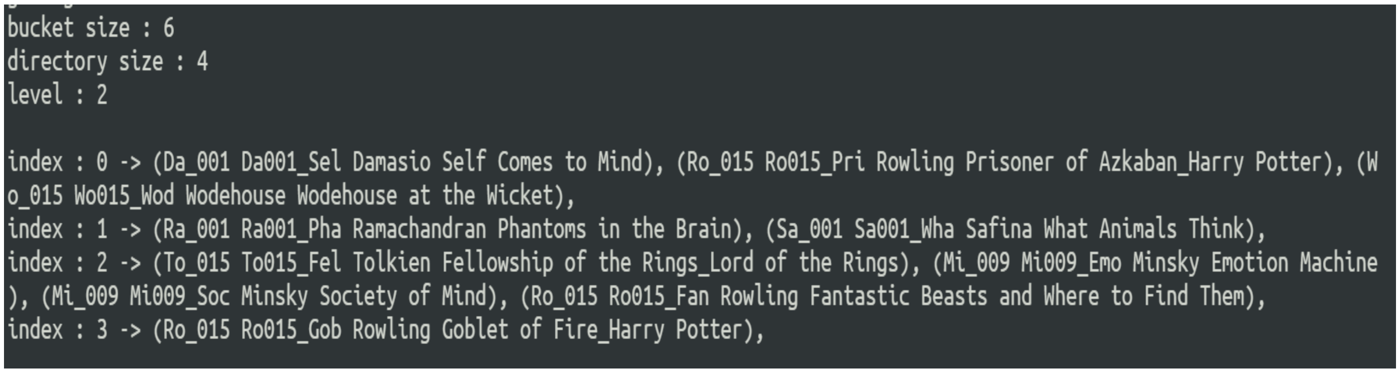
For bucket size 2



```
bucket size : 2
directory size : 8
level : 3

index : 0 -> (Da_001 Da001_Sel Damasio Self Comes to Mind), (Ro_015 Ro015_Pri Rowling Prisoner of Azkaban_Harry Potter),
NULL
index : 2 -> (To_015 To015_Fel Tolkien Fellowship of the Rings_Lord of the Rings), (Mi_009 Mi009_Emo Minsky Emotion Machine
),
index : 3 -> (Ro_015 Ro015_Gob Rowling Goblet of Fire_Harry Potter),
index : 4 -> (Wo_015 Wo015_Wod Wodehouse Wodehouse at the Wicket),
index : 5 -> (Ra_001 Ra001_Pha Ramachandran Phantoms in the Brain), (Sa_001 Sa001_Wha Safina What Animals Think),
index : 6 -> (Mi_009 Mi009_Soc Minsky Society of Mind), (Ro_015 Ro015_Fan Rowling Fantastic Beasts and Where to Find Them),

NULL
```

Clearly visible that since our bucket size is less so we got high number of levels in the directory

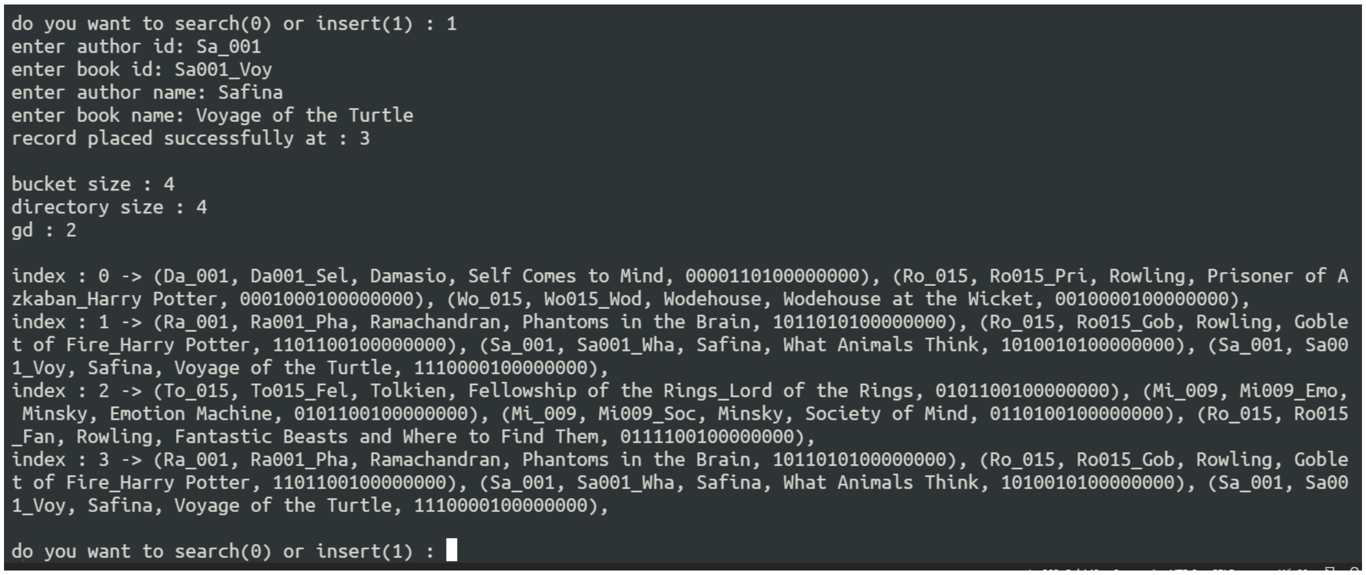
For bucket size 6



```
bucket size : 6
directory size : 4
level : 2

index : 0 -> (Da_001 Da001_Sel Damasio Self Comes to Mind), (Ro_015 Ro015_Pri Rowling Prisoner of Azkaban_Harry Potter), (W
o_015 Wo015_Wod Wodehouse Wodehouse at the Wicket),
index : 1 -> (Ra_001 Ra001_Pha Ramachandran Phantoms in the Brain), (Sa_001 Sa001_Wha Safina What Animals Think),
index : 2 -> (To_015 To015_Fel Tolkien Fellowship of the Rings_Lord of the Rings), (Mi_009 Mi009_Emo Minsky Emotion Machine
), (Mi_009 Mi009_Soc Minsky Society of Mind), (Ro_015 Ro015_Fan Rowling Fantastic Beasts and Where to Find Them),
index : 3 -> (Ro_015 Ro015_Gob Rowling Goblet of Fire_Harry Potter),
```

Ans 6)

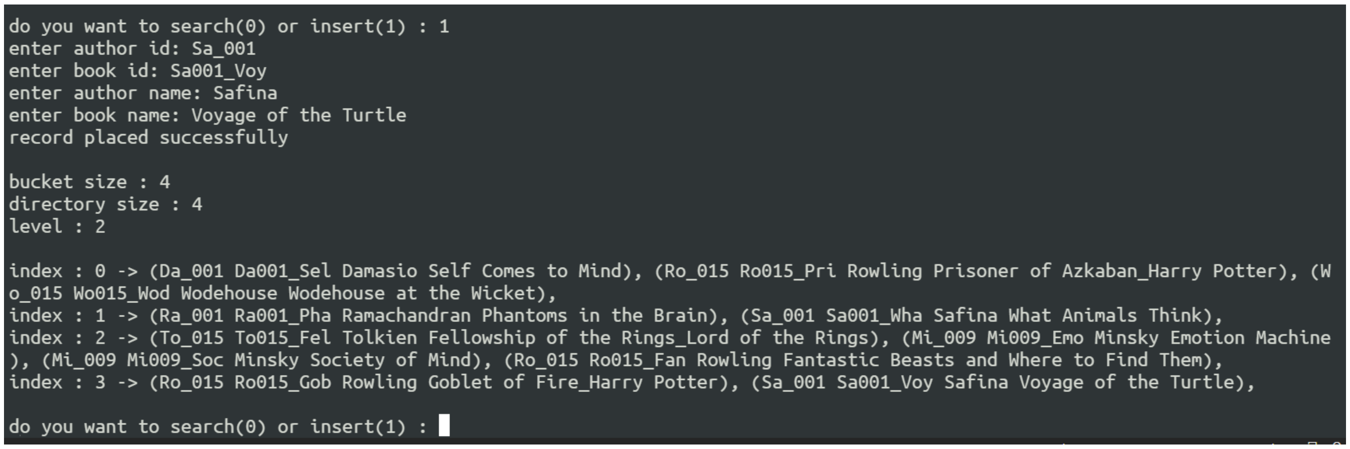
Bucket size is: 4
For extendible hashing



```
do you want to search(0) or insert(1) : 1
enter author id: Sa_001
enter book id: Sa001_Voy
enter author name: Safina
enter book name: Voyage of the Turtle
record placed successfully at : 3

bucket size : 4
directory size : 4
gd : 2

index : 0 -> (Da_001, Da001_Sel, Damasio, Self Comes to Mind, 0000110100000000), (Ro_015, Ro015_Pri, Rowling, Prisoner of A
zkaban_Harry Potter, 0001000100000000), (Wo_015, Wo015_Wod, Wodehouse, Wodehouse at the Wicket, 0010000100000000),
index : 1 -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Ro_015, Ro015_Gob, Rowling, Goble
t of Fire_Harry Potter, 1101100100000000), (Sa_001, Sa001_Wha, Safina, What Animals Think, 1010010100000000), (Sa_001, Sa00
1_Voy, Safina, Voyage of the Turtle, 1110000100000000),
index : 2 -> (To_015, To015_Fel, Tolkien, Fellowship of the Rings_Lord of the Rings, 0101100100000000), (Mi_009, Mi009_Emo,
 Minsky, Emotion Machine, 0101100100000000), (Mi_009, Mi009_Soc, Minsky, Society of Mind, 0110100100000000), (Ro_015, Ro015
_Fan, Rowling, Fantastic Beasts and Where to Find Them, 0111100100000000),
index : 3 -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Ro_015, Ro015_Gob, Rowling, Goble
t of Fire_Harry Potter, 1101100100000000), (Sa_001, Sa001_Wha, Safina, What Animals Think, 1010010100000000), (Sa_001, Sa00
1_Voy, Safina, Voyage of the Turtle, 1110000100000000),

do you want to search(0) or insert(1) :
```

We can clearly see that is is stored in the 3rd index and no overflowing happens

For linear hashing



```
do you want to search(0) or insert(1) : 1
enter author id: Sa_001
enter book id: Sa001_Voy
enter author name: Safina
enter book name: Voyage of the Turtle
record placed successfully

bucket size : 4
directory size : 4
level : 2

index : 0 -> (Da_001 Da001_Sel Damasio Self Comes to Mind), (Ro_015 Ro015_Pri Rowling Prisoner of Azkaban_Harry Potter), (W
o_015 Wo015_Wod Wodehouse Wodehouse at the Wicket),
index : 1 -> (Ra_001 Ra001_Pha Ramachandran Phantoms in the Brain), (Sa_001 Sa001_Wha Safina What Animals Think),
index : 2 -> (To_015 To015_Fel Tolkien Fellowship of the Rings_Lord of the Rings), (Mi_009 Mi009_Emo Minsky Emotion Machine
), (Mi_009 Mi009_Soc Minsky Society of Mind), (Ro_015 Ro015_Fan Rowling Fantastic Beasts and Where to Find Them),
index : 3 -> (Ro_015 Ro015_Gob Rowling Goblet of Fire_Harry Potter), (Sa_001 Sa001_Voy Safina Voyage of the Turtle),

do you want to search(0) or insert(1) : 
```

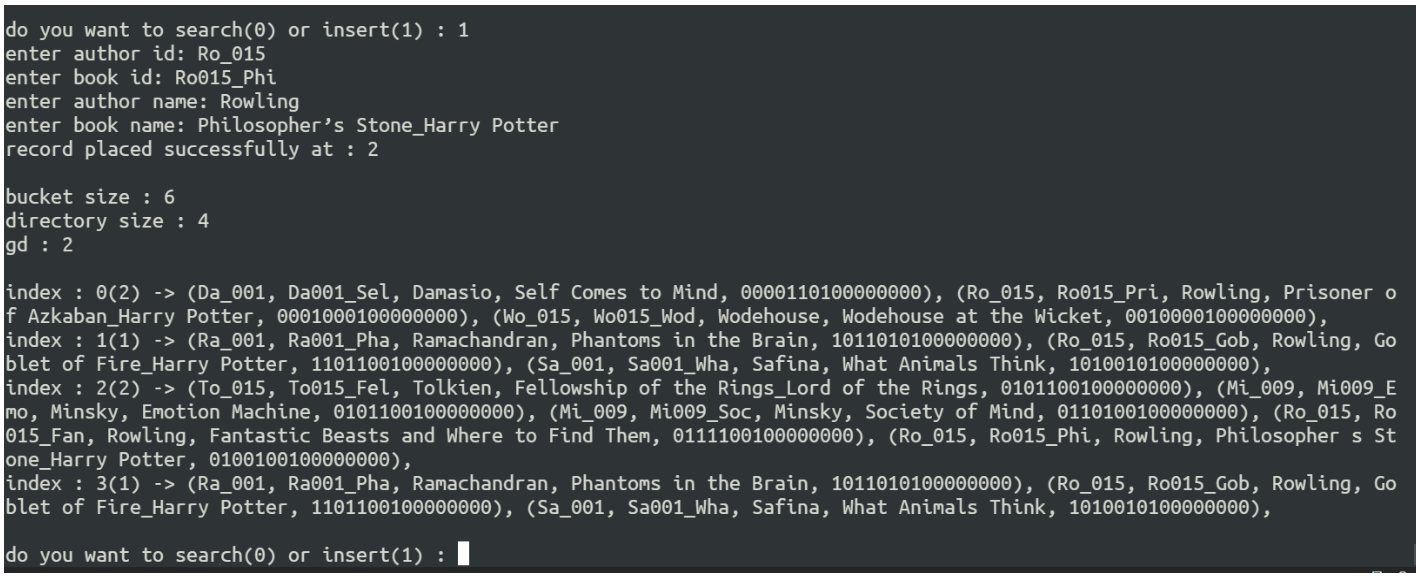
Here also it is inserted at location 3 and it is a mere coincidence that it is inserted same as above but we can see the rest of the elements are differently placed in the directory this is due to the algorithms that are implemented behind

In this case if we see the insertion in ext Hashing algorithm it is in O(1) since no overflow happens here so no need to alter the pointer or increase the size of the directory and in case of linear hashing also the insertion is in O(1) as no overflow occurs so no change in pointers just insert the data there now the difference between them will occur when we will have an overflow so in case there is an overflow in ext hashing algorithm it will check for the local directory and global directory if (ld < gd) then go for split or otherwise go for size increasing the directory size and then split but on the other hand in linear hashing if overflow occurs then rehash the directory and the insert the required element

Now coming to the search time so we can see that in the code that it is mere O(size of bucket filled) since the bucket size is constant so it is also a constant time so searching in both the algorithms is constant time

Ans 7)

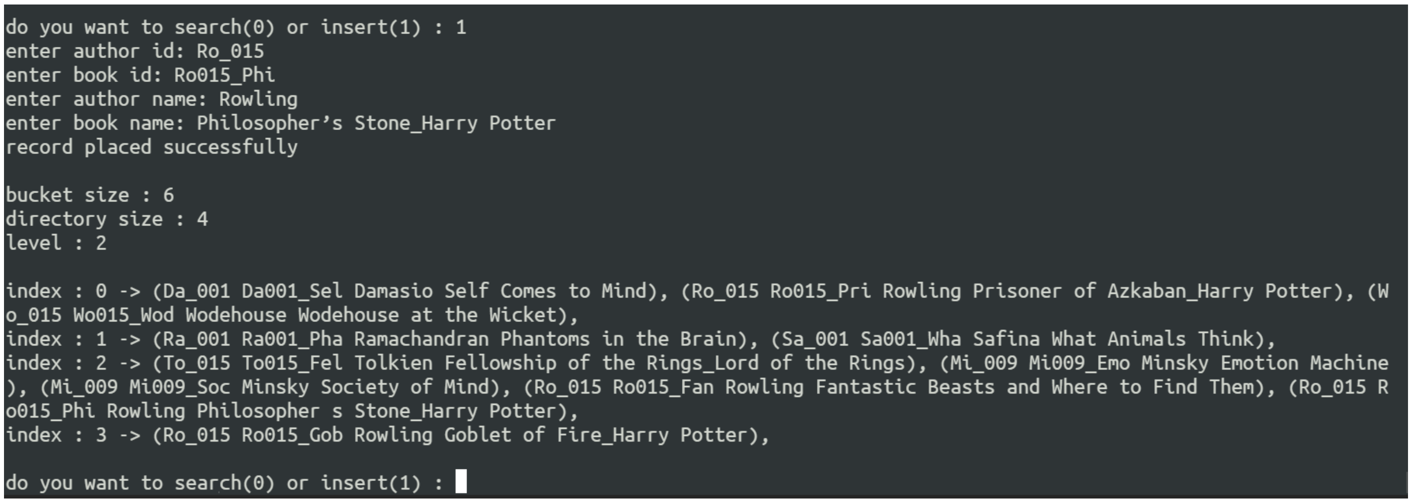
Bucket size : 6
For extendible hashing



```
do you want to search(0) or insert(1) : 1
enter author id: Ro_015
enter book id: Ro015_Phi
enter author name: Rowling
enter book name: Philosopher’s Stone_Harry Potter
record placed successfully at : 2

bucket size : 6
directory size : 4
gd : 2

index : 0(2) -> (Da_001, Da001_Sel, Damasio, Self Comes to Mind, 0000110100000000), (Ro_015, Ro015_Pri, Rowling, Prisoner o
f Azkaban_Harry Potter, 0001000100000000), (Wo_015, Wo015_Wod, Wodehouse, Wodehouse at the Wicket, 0010000100000000),
index : 1(1) -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Ro_015, Ro015_Gob, Rowling, Go
blet of Fire_Harry Potter, 1101100100000000), (Sa_001, Sa001_Wha, Safina, What Animals Think, 1010010100000000),
index : 2(2) -> (To_015, To015_Fel, Tolkien, Fellowship of the Rings_Lord of the Rings, 0101100100000000), (Mi_009, Mi009_E
mo, Minsky, Emotion Machine, 0101100100000000), (Mi_009, Mi009_Soc, Minsky, Society of Mind, 0110100100000000), (Ro_015, Ro
015_Fan, Rowling, Fantastic Beasts and Where to Find Them, 0111100100000000), (Ro_015, Ro015_Phi, Rowling, Philosopher s St
one_Harry Potter, 0100100100000000),
index : 3(1) -> (Ra_001, Ra001_Pha, Ramachandran, Phantoms in the Brain, 1011010100000000), (Ro_015, Ro015_Gob, Rowling, Go
blet of Fire_Harry Potter, 1101100100000000), (Sa_001, Sa001_Wha, Safina, What Animals Think, 1010010100000000),

do you want to search(0) or insert(1) :
```

Here it is inserted at 2 and since the bucket size is large so no overflow

Bucket size : 6
For linear hashing



```
do you want to search(0) or insert(1) : 1
enter author id: Ro_015
enter book id: Ro015_Phi
enter author name: Rowling
enter book name: Philosopher’s Stone_Harry Potter
record placed successfully

bucket size : 6
directory size : 4
level : 2

index : 0 -> (Da_001 Da001_Sel Damasio Self Comes to Mind), (Ro_015 Ro015_Pri Rowling Prisoner of Azkaban_Harry Potter), (W
o_015 Wo015_Wod Wodehouse Wodehouse at the Wicket),
index : 1 -> (Ra_001 Ra001_Pha Ramachandran Phantoms in the Brain), (Sa_001 Sa001_Wha Safina What Animals Think),
index : 2 -> (To_015 To015_Fel Tolkien Fellowship of the Rings_Lord of the Rings), (Mi_009 Mi009_Emo Minsky Emotion Machine
), (Mi_009 Mi009_Soc Minsky Society of Mind), (Ro_015 Ro015_Fan Rowling Fantastic Beasts and Where to Find Them), (Ro_015 R
o015_Phi Rowling Philosopher s Stone_Harry Potter),
index : 3 -> (Ro_015 Ro015_Gob Rowling Goblet of Fire_Harry Potter),

do you want to search(0) or insert(1) :
```

Here the tuple is saved at index 2 since the value of value of bucket size is 6 which is large enough to handle this element to be inserted

In this case if we see the insertion in ext Hashing algorithm it is in O(1) since no overflow happens here so no need to alter the pointer or increase the size of the directory and in case of linear hashing also the insertion is in O(1) as no overflow occurs so no change in pointers just insert the data there now the difference between them will occur when we will have an overflow so in case there is an overflow in ext hashing algorithm it will check for the local directory and global directory if (ld < gd) then go for split or otherwise go for size increasing the directory size and then split but on the other hand in linear hashing if overflow occurs then rehash the directory and the insert the required element

Now coming to the search time so we can see that in the code that it is mere O(size of bucket filled) since the bucket size is constant so it is also a constant time so searching in both the algorithms is constant time

Ans 8)

To encode author id or book id we can use the following strategies like in the above example that we have taken in the table there our book_id is a function of author_id and book(first three letter) but here is an issue with that since one author can write infinite books which have same first three letter so we need to think of some other way to compute the book_id so one way that we can think here is since the author_id is different for different authors so this part is not an issue but the issue is the first three letters of the book due to which the book_id is not unique in the entire table to solve it we can have our book_id as authorid_bookno where bookno is the number that is assigned to the book that author has written so this number is equal to the number of books written by author so far + 1
For examples if author has written a book which is its 5th book which is going to be published so we will have our book_id as author_id_5 since it is its 5th book in this way even if the author id is same for a book but the number that is assigned to the book will be different surely so in this way book_id will be unique
So that’s why my solution gives a more concrete and unique book_id

Bonus question:
If we were to implement linked list then we can implement hashing with chaining in which there is global directory and if at any point overflow occurs then we can just extend the value of that particular index using linked list and so it feels like a chain is created

For this implementation the code is president in the zip file

Insertion time : O(size of string to be put into the bucket) it is size of string because of hash function which will convert the string to the index to be placed in the list
Search time : O(1+size of the linked list at hashed location)

Here if the global directory size is increased then we got benefit since our bucket size is dynamic so if we increase our global directory size then it reduces the chances of collisions
And make our search time even faster