Report of Database Lab3
Name: Nayun Xu
Student ID: 515030910635

1.Design decisions

1.1 HashEquiJoin

I use a ***HashMap*** to store the elements of the child that has more tuples. And iterate through the other child. Use an ***ArrayList*** to store the elements with same the key field. I didn't add limit to the size of ***HashMap,*** so it cannot be applied to large scale data.

2.Changes on API

No changes.

3.Incomplete elements

I still don't know how to implement the lock. The size limit of ***HashMap*** in ***HashEquiJoin.***

4.Experience

I spent about 10 to 15 hours(not exact numbers) on this lab. I first face difficulties when implementing ***HashEquiJoin***. I didn't know what to do in this part, and I used google to solve this problem. And the second difficult part is the insert and delete in ***HeapFile***. I first tried the same structure as what I did in BTreeFile. But I found it didn't work when adding new pages into the file. I just add new pages into the dirty cache and return it back to the ***BufferPool***, which means nobody is taking care of extending the heapfile length. This is because in ***BTreeFile***, the ***createEmptyPage*** method has done this part for me. And also, the adding all pages that are accessed into the dirty cache is not necessary in ***HeapFile***, because only one page will be modified, which is different to ***BTreeFile*** where the leafpage's ancestors may be modified. So finally I modified the structure into a much simpler one. Another problem I encountered is after inserting a tuple, I should modify its ***recordId*** to match its location, which was missed when I first implemented this part.