student:Wang Yan
student ID:2013950450

Question 1

First of all ,we need calculate several important parameters.
As we know,the records of R and S are both 250 bytes,and one block is 4096 bytes,as 4096/250=16.3384=16,which means that each block could store 16 records

As a result,
for R,it need 20,000/16=1250 blocks
for S,it need 4,000/16=250 blocks

(a)

(a1)simple nested loops join

Depend on the formula,
when R was outer relation
Cost=1250+20,000*250=5,001,250

When S was outer relation
Cost=250+4,000*1250=5,000,250

Obviously,we will take S as our outer relation

Some ideas about the optimizing of this method.
As we know S.b is the primary key for S,which is in order and unique.
As a result,when we took S as inner relation,we can stop when R.a=S.b.

```
for each R.a
      for each S.b
            if R.a=S.b
            break;
```

It will reduce the operations

(a2)block nested loops join

Depend on the formula, when R was outer relation,As we want the minimus IO consuming and buffer blocks is 52.
So we load 50 blocks of R,and left 1 blocks for S,1 for output.
Then the Cost= 1250+(1250/50)*250=7500

Similarly,when the S was outer relation
Then the Cost= 250+(250/50)*1250=250+5*1250=6500

So we take S as outer relation

(a3)merge join

As we know,the S is primary key,so we needn't to sort it.

Then for R,we need to sort it,but we only have 52 blocks buffer,so we need use external sort-merge algorithms.
Depend on its formula,Sorting Cost=1250*(2(log51(1250/52))+1)=1250*(2+1)=3750
Then based on the merge join formula,the total Cost=3750+0+1250+250=5170;

(b)

As we know,the total Cost=(blocks of outer relation)+(tubles of outer

relation)*(index search cost of inner relation);And both R and S has B+-tree indexes.
Now,we want to calculate the search cost of inner relation.
For a B+-tree, each node is a block.
We assume the degree of B+ is n,then 64+8+(8+8)*(n-1)=4096 bytes,then the n=251
So for the non-leaf node ,it has up to 251 points. For the leaf node, it has up to 250 entries.
Then depend on the definition of the the B+ tree.
The height of the R equal to 1+upperBound(log251(20,000/250))=2;

However,I have another method to solve this problem.

When R was the outer relation.
The leaf nodes of R is 20,000/(251-1)=80 leaf nodes. As we know ,the non-leaf node can have up to 251 pointers,which means it have up to 251 leaf nodes.
So we only have two level for this B+ tree,which means the search cost is 2.
Total Cost=20000+20000*2=60000;

When S was the outer relation.
The leaf nodes of S is 4,000/250=16 leaf nodes
Similarly,we only have two level for this B+ tree,the search cost is 2.
Total Cost=4000+4000*2=12000

So the minimus cost is 12000

(c)

For index nested loops, depend on its formula.
We set the number of available buffer as M.
Depend on the formula Cost=block of outer relation+tubles of outer relation*inner index search cost.

Obviously,the block of outer relation and tubles of outer relation are not effected by the buffer number.
And the inner index search cost is decided by the size of the buffer block, not the number of the buffer.
So the buffer number will not affect the result.

For block nested loops join.
Depend on the formula Cost= block of outer relation+(upperBound(block of outer relation/buffer number-2))*blocks of inner relation.
It is manifest that the larger of the buffer,the smaller of the Cost.
However,when the buffer is larger than the ourer relation blocks,the Cost=the blocks of outer relaition+the blocks of inner relation, and have no business with the buffer.

Question 2

For plan (a)

The Cost of filter the S =100+20=120;
The T2 has 2000 tuples and 20 blocks.
Then we jion T2 and R by nested loops jion,and R as outer relation.
By its formula,the Cost=100+10000*20=200100;

For plan(b)

It use index nested loop jion,and we can get form the quesion that the B+ tree is 3 levels.
It means that the search cost of B+-tree is 3,and R as outer relation.
By the formula, we can calculate the Cost=10000+10000*3=40000;

It is clear that plan(b)is better;