CISC637 HW4
Hao Guo
702311031

P1.
a)# of records per block = 4096 / 256 = 16
file size = 4000 / 16 = 250 block

b)
# of records per block: 4096 / (8+8) = 256
file size: 4000/256 = 16 block

c)
records/ block = 4096 / 30 = 136.53 = 136
file size = 3500000 / 136 = 25735.29 = 25736

d)
total bytes: 8 + 4 + 6 + 4 + 8 = 30 bytes
# of records per block = 4096 / 30 = 136 lower bound
file size = 96000 / 136 = 706 block (upper bound)

e)
each size is 4 + 35*8 = 284
# of records per block = 4096/(4 + 35*8) = 14
file size = (3500000 / 35) * (4 + 35*8) = 28,400,000
total blocks: 28,400,000 / 4096 = 6934 blocks

P2.
a)
(4000*256 / 4096) * 1ms = 250ms

b)
$(O(\log_2 16) + 1) * 1ms = 5ms$

c)
the # of blocks for records that match the query is 100 / 16 = 6.25 = 7
$(O(\log_2 16 + 7)) * 1ms = 11ms$

d)
this one needs the full file scan and the file size is 25736 block.
25736 * 1ms = 25736ms

e)

we use the primary index search and the index scan has 706 blocks and they may match records about 1 or 2 blocks, so the time should be:

$O(\log_2 B_1 + m) = (\log_2 706) + 1 = 10 + 1 = 11ms$
$O(\log_2 B_1 + m) = (\log_2 706) + 2 = 10 + 2 = 12ms$

f)
$\log_2((3500000/35) * 4 / 4096)$ ms + 1ms (since 35* 8 = 280 < 4096, and the pointers are in the same block) + 3ms (in the worst case that 40 students' records are in 3 different blocks) = 7 + 1 + 3 = 11ms

P3.
a). find records with search-key value 23.
First root node 11 → right node 19 → right node 23 → second node 23 → records of node 23.
Totally need 5 block reads.

Find record between 11 and 23.
Find record between 11 and 23. Root node 11 --> right node 19 --> left node 17 --> left node 11 --> left node 17 --> left node 19 --> left node 23. (this contains 7 block reads) + additional 4 more records read (11 , 17 , 19 , 23). Total = 7 + 4 = 11.
Totally need 11 block reads.

Find records with search-key values less than 7.

The steps of reading the record less than 7 is root node 11 → left node 5 → left node 3 → left node 2 → left node 3 → left node 5 → left node 7 (7 blocks)
And we need 3 more blocks to read, so the total # of block is 7+3 = 10.

b) find records with search-key value 23.

Since the height of the B+ tree is 2 and the steps of reading the record with search value 23 is: right root node → left node 23 → records of 23.

Total blocks read: 3

Find records between 11 and 23.

Since the height of B+ tree is 2 and steps are right root node 11 → left node 11 → left node 23 and then read 4 more blocks have the records.
Total step: 7

Find values less than 7.

Since the height of the B+ tree is 4 and the steps of reading the record less than 7 is left root node 5 → left node 2 → left node 7 (3 blocks) and 3 more blocks to read the record.
Total step: 6

P4.
Estimate the number of block reads required for each of the following operations.
1 since this one use the select * from R1 where b = 1545, this required the whole blocks read, so the answer should be (10000/ 200) = 50 blocks read.

2 it required $h_c + 1$ operations to find that c= 2883, and the total block reads is 3 + 1 = 4

3 it required $h_d + 1$ operations to find that d= 521 and we have 10 records with d = 521, and the total block reads is 2 + 10 = 12 blocks

4 indexed nested-loop join: since R2.c that $h_c = 3$ and R1 to R2 they are 1 to 1 relation. So the block reads should be: 10000/ 20 + 10000(3 + 1) = 40050

5 block nested-loop join: this join operation needs iterate every pair blocks of R1 and R2, so the block reads is : 10000/200 + (10000/200) * (45000/2250) = 1050

6 sort-merge join:

$2*B_2 * \log_2 B2 + B1 + B2 = 2*50 * \log_2 50 + 50 + 20 = 100* 6 + 70 = 670$ block reads.

P5.
a)this one if we use full file scan total block is 5000000/ 10 = 500000, but if we chose the first primary B+ tree on a, then the total steps are : $h_a + 1 = 4+1 = 5$. The # of blocks we need to read is 5 and the estimate records we need to read and the # is greater than 5.
So we should choose the 1st primary index.

b)we should choose 3 hash index (hash index on c with 50 buckets), since with this index and each buckets have 100000 index records and 100 index per block so the # of blocks are 100000 / 100 = 1000, the total blocks read : 1000 + 1= 1001 .

c)

We should choose 3 the hash index on c with 50 buckets, (5000000 / 50) / 100 = 1000 block and the estimate number of block we need to read are 1000 + 1 = 1001, and if we use the 1st primary index the block we need to check is 1000 + 4 = 1004, which is greater than 1001 on hash index on c with 50 buckets.
So the total block reads is: 1000 + 1 = 1001
And we chose 3rd hash index on c.

d)

we choose 2nd index on b.
Since there are 100000 unique value in b and total 5000000 records in the table.
So the average duplicate number of each unique b is 50. And the average case is that they are sorted in 50 blocks.
So for b = 60155.76 stored in 50 blocks. Then if we use secondary index on b need 2 blocks to find the index and 50 blocks to find the result.
Totally it would be 52 blocks.
We choose secondary index on b.

e) this one since the relation is OR, so we need to consider all the possible conditions, We should choose 3 the hash index on c with 50 buckets, (5000000 / 50) / 100 = 1000 block, the total step is: 1000 + 1 = 1001. And the total step for B+ tree index on a: 4 + 100 + 1000 = 1104

And the final step should be 1st index + 3rd index, the total: 4 + 100 + 1000 + 1000 + 1 = 2105

P6.
1 explain in words what each of the following relational algebra expressions is requesting from the university database. Write SQL queries that are equivalent to each one.

A it shows the course title from the course that the course's department name is Physics and credits is greater or equal to 4.

SELECT Course.title FROM Course where course.credits >= 4 AND dept.name = 'Physics';

B, it shows the student name and student ID from table student and takes and have the condition that course ID is CISC437.

SELECT student.ID, student.name FROM takes natural join student where course_id ='cisc437';

C, it shows the student ID and student name and the takes ID that the course_id is CISC437.

SELECT ID, name FROM takes natural join student where takes.course_id = 'CISC437';

D, Count the average open seats of courses that were taught in this semester, and sort them by the department name.
SELECT dept_name,AVG(capacity) FROM classroom c natural join section s natural join course GROUP BY dept_name;

Write SQL queries and relational algebra expressions for the following questions relating to the university.

A, SELECT Student.name FROM student WHERE Student.dept_name = 'Computer Science'.

$\prod$name ($\sigma dept_{name} = 'Computer\ Science'(student)$)

B, SELECT DISTINCT name FROM takes NATURAL JOIN course NATURAL JOIN section NATURAL JOIN student WHERE course.dept_name = ‘Physics’;

$$\pi_{name}(\sigma_{\text{course.dept\_name='Physics'}}(takes \bowtie \text{course} \bowtie \text{section} \bowtie \texttt{student})$$

C,
select distinct course_id from teaches where course_id in (select course_id from teaches where semester = 'Spring' and year = 2011) and course_id in (select course_id from teaches where semester ='Fall' and year =2012);

$$(\pi_{\text{course\_id}}(\sigma_{\text{semester}='\text{Spring}'\wedge\text{year}='2011'}(\text{teaches})) \cap (\pi_{\text{course\_id}}(\sigma_{\text{semester}='\text{Fall}'\wedge\text{year}='2012'}(\text{teaches}))$$

D, SELECT dept_name, avg(salary) FROM instructor GROUP BY dept_name;

$$_{\text{dept\_name}}G_{\text{average(salary)}}(\text{instructor})$$

E,
select i.name, count(s.name) from student s join advisor a on s.ID = a.s_id join instructor i on a.s_id = i.ID where i.dept_name = 'computer science' group by i.name;

$$_{\text{instructor.name}}G_{\text{count(student.name)}}(\sigma_{\text{student.ID=advisor.s\_id} \wedge \text{advisor.s\_id = instructor.ID} \wedge \text{dept\_name = ‘computer science’}}(\text{student} \bowtie \text{advisor} \bowtie \text{instuctor}))$$