1.

1. a) Ceiling(10000/3) =3334 sorted runs
   b) Ceiling(20000/3)=6667 sorted runs
   c) Ceiling(2000000/3)=666667 sorted runs
2. Formula used is $1 + \lceil \log_{B-1} K \rceil$ where K is Ceiling (N/B)
   a) 13 passes
   b) 14 passes
   c) 21 passes
3. Formula used is 2*N*Number of passes
   a) 260000 I/Os
   b) 560000 I/Os
   c) 84000000 I/Os
4. Formula used is B-1 >= Ceiling (N/B)
   a) 101 buffer pages
   b) 142 buffer pages
   c) 1415 buffer pages

2.

1. Conjunctive normal form is a group of conjuncts combined using the and operator. Conjunctive normal form is important in the context of relational query evaluation because in a CNF expression indexes often exist over some subset of conjuncts. Often indexes are used to increase the selectivity of operator by doing selection over multiple conjunctions using a single multi-attribute index. This is possible as the order of the conjuncts does not matter in a CNF expression.

2. First, sort the tuples using the GROUP BY attribute.The second step is to re-sort every group by sorting all elements on the MAX attribute. In the second step do not re-sort beyond the group boundaries.

3. Sorting can be used to implement SECOND LARGEST. For each group, if has a GROUP BY clause, sort the tuples and return the desired attribute that is second largest for each group. The cost of this algorithm is the cost of sorting.

3. The source for question three is “Fall 2004 CS 186 Solutions”, 2004.

1. The relational algebra expression is as follows:
   π D.dname, F.budget ((π B.did (σ E.sal >= 59000, E.hobby =”yodelling” (E)) ⋈ π D.did, D.dname (σ D.floor = 1 (D))) ⋈ π F.budget, Fdid (F))
2. The following 6 join plans are possible when only considering left deep joins:
   1. ((D E) F)
   2. ((E D) F)
   3. ((D F) E)

4. ((F D) E)
5. ((E F) D)
6. ((F E) D).

In the where clause, there are two join conditions: E.did = D.did, D.did = F.did. This means (E join D) and (D join F). Although, E join F is equivalent to a cross product so we can discard the last two plans listed. To simplify, the permutations between 1-2 and 3-4 can be ignored. Therefore there are two join orders ((D E) F) and ((D F) E)

3. i)

Emp has an initial cardinality of 50 000. It has selection condition E.sal >= 59 000 and E.hobby = 'yodelling'. To find the resulting cardinality the factors with respect to salary and hobbies must be found. It is given that the max salary is 60 000, the min salary is 10 000, and the selection condition is E.sal >= 59 000. Therefore the salary factor is (60 000 - 59 000)/(60 000 - 10 000) = 0.02. It is given that 200 unique hobbies exist. Therefore the factor for hobbies is 0.005 (1/200). Therefore the resulting cardinality is 50000 * 0.02 * 0.005 which is 5.

Dept has an initial cardinality of 5000. It has the selection condition D.floor = 1. It is given that the company owns two floors in the building. Therefore the resulting cardinality is 5000 * 0.5 = 2500.

Finance has an initial cardinality of 5000. There is no selection condition therefore the resulting cardinality is 5000

ii)

First, the cost of the ((E D) F) join is considered.

1. There will be no temporary relations created as the tuples from E will be pipelined.
2. The tuple with E.salary >= 59 000 can be retrieved using the B+ index.
3. There are no B+ tree indexes that exist of E.hobby so preselection with respect to E-hobby is not possible.
4. It is estimated that 1000 (50000*0.2) tuples with E.salary >= 59 000 are selected. This costs 1 tree traversal (estimated 3 I/Os) + the cost of scanning the leaf pages 5 (1000/200) + the retrieval cost for the 1000 tuples (each tuple is potentially 1 disk I/O as the index is unclustered). This is equal to 1008 (3+5+1000).
5. Do an on-the-fly select of the 100 retrieved tuples to find those with E.hobby = "yodelling". There is estimated to be 5 such tuples (1000/200).
6. Join the result with D. The resulting tuples from step 5 are pipelined from E one at a time to D. Use the fact that D.did is a key and there is a B+-tree index on D.did to find the matching tuples for the join. This is done by searching the D.did B+-tree and, as D.did is a key, this will retrieve at most one matching tuple per tuple from E.
7. The cost of the index nested loop is the cost of E join D. This is 5*(tree traversal of D.did B+-tree + record retrieval)= 5*(3+1)=20.
8. The above step will lead to at most 5 tuples as, since D.did is a key, it it expected to only have 1 matching D tuple for each E tuple.

9. Select the tuples that have D.floor =1 on the fly and pipeline it to the next level F. Estimated to result in 3 tuples (5/2).
10. Use the facts that F.did is a key and there is a B+-index on F.did to retrieve at most 1 tuple for each of the 3 pipelined tuples. The cost of this is 3*(tree traversal of F.did B+-tree + record retrieval)=3(3+!)=12.
11. The total cost, ignoring the cost of writing out the final result, is 1040 (1008+20+12).

In the other join orders ((D E) F), (D F) E), and ((F D) E) the FD join will result in a high initial cost. In part B it was calculated that the D join F results in an initial tuple retrieval cost of 2500 for D and 5000 for E. This is higher the the cost calculated for the ((E D) F) order.

A ((D E) F) join order causes an initial tuple retrieval cost of 2500 due to selecting from D first. Again, this is higher the the cost calculated for the ((E D) F) order. Therefore ((E D) F) is the best join order.