1. Consider a schema $R(Id_R, A, B, \ldots, IdS*)$, $S(IdS,\ldots)$, $T(IdT, C, D,\ldots, IdS*)$ and the following query

```
SELECT DISTINCT T.C count(*)
FROM R, T
WHERE R.IdS = T.IdS And R.A \leq 100 And T.D = 10
GROUP BY T.C, T.D
```

Assume that $R$ is stored with a dense clustered index on $A$, while $S$ and $T$ are stored as heap files. Primary keys are $R.IdR$, $S.IdS$ and $T.IdT$, while $R.IdS$ and $T.IdS$ are foreign keys that refers to $S$. Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, and on $T.D$. Assume that the size of all indexes only depend on the number of RIDs, as indicated in the table below. Assume that every page is 4.000 bytes long, and that every attribute uses 4 bytes. Assume a buffer size of 100 pages. If you need Cardenas formula $\Phi(n,k)$, approximate it with $\min(n,k)$.

<table>
<thead>
<tr>
<th></th>
<th>NReg (R)</th>
<th>NPag (T)</th>
<th>NLeaf of Indexes (S)</th>
<th>NKey (R)</th>
<th>Min (R)</th>
<th>Max (R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>500,000</td>
<td>10,000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>100,000</td>
<td>2,000</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>10,000,000</td>
<td>50,000</td>
<td>20,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx.R.A</td>
<td>See R</td>
<td>100</td>
<td>0</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx.T.D</td>
<td>See T</td>
<td>100</td>
<td>0</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) Is DISTINCT redundant?
b) Compute the selectivity factor of the three predicates in the condition
c) Compute the cost of accessing $\sigma_{A \leq 100}(R)$, $\sigma_{D \leq 10}(T)$ (useless for this query) and $\sigma_{D=10}(T)$, with and without index. Remember that the index on $R.A$ is clustered.
d) Compute the cost of a MergeJoin plan that uses the cheapest plans for the selections (ignore the fact that $IdS$ is not a key for neither table) (draw the plan first)
e) Compute the cost for an IndexNestedLoop plan where $R$ is the outer (left-hand-side) leaf (draw the plan first)

2. a) Assume two relation $R(a,b)$ and $S(a,c)$. Define the left natural outer join($R,S$) (leftjoin($R,S$)) and give a very simple example.
b) Write a two-lines (more or less) pseudo code specification for IndexNestedLoop. Briefly specify (using natural language or pseudo-code) how one could generalize IndexNestedLoop to LeftIndexNestedLoop in order to compute left outer join
c) Compare the cost of LeftIndexNestedLoop($OE, OI, condition$)) with the cost of IndexNestedLoop($OE, OI, condition$)

3. a) Explain (briefly) why heap organization is the one that is most commonly used
b) In which situations the heap organization is the best one?
c) In which situations is the hash (procedural) organization the best one?
1. Consider a schema \( R(\text{IdR}, A, B, \ldots, \text{IdS}^*), S(\text{IdS}^*), T(\text{IdT}, C, D, \ldots, \text{IdS}^*) \) and the following query

\[
\text{SELECT DISTINCT T.C} \quad \text{count(*)}
\]

\[
\text{FROM} \quad R, T
\]

\[
\text{WHERE} \quad R.\text{IdS} = T.\text{IdS} \text{ And } R.A \leq 100 \text{ And } T.D = 10
\]

\[
\text{GROUP BY} \quad T.C, T.D
\]

Assume that \( R \) is stored with a dense clustered index on \( A \), while \( S \) and \( T \) are stored as heap files. Primary keys are \( R.\text{IdR}, S.\text{IdS} \) and \( T.\text{IdT} \), while \( R.\text{IdS} \) and \( T.\text{IdS} \) are foreign keys that refers to \( S \). Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, and on \( T.D \). Assume that the size of all indexes only depend on the number of RIDs, as indicated in the table below. Assume that every page is 4,000 bytes long, and that every attribute uses 4 bytes. Assume a buffer size of 100 pages. If you need Cardenas formula \( \Phi(n,k) \), approximate it with \( \min(n,k) \).

<table>
<thead>
<tr>
<th></th>
<th>NReg</th>
<th>NPag</th>
<th>NLeaf of Indexes</th>
<th>NKey</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R )</td>
<td>500.000</td>
<td>10.000</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( S )</td>
<td>100.000</td>
<td>2.000</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T )</td>
<td>1,000.000</td>
<td>50.000</td>
<td>2.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{Idx.R.A} )</td>
<td>See ( R )</td>
<td>100</td>
<td>0</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{Idx.T.D} )</td>
<td>See ( T )</td>
<td>100</td>
<td>0</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) Is DISTINCT redundant?

Yes, it is, since the closure of the projected attribute \( T.C \) is \{\( T.C, T.D \}\), which includes the GROUP BY attributes

b) Compute the selectivity factor of the three predicates in the condition

\[
s_f(R.\text{IdS} = T.\text{IdS}) = 1/\text{NReg}(S) = 1/100.000
\]

\[
s_f(R. A \leq 100) = 100/\text{Max}(R.A) = 100/1.000 = 1/10
\]

\[
s_f(T.D = 10) = 1/\text{NKey}(T.D) = 1/100
\]

c) Compute the cost of accessing \( \sigma_{A \leq 100}(R) \), \( \sigma_{D \leq 10}(T) \) (useless for this query) and \( \sigma_{D = 10}(T) \), with and without index. Remember that the index on \( R.A \) is clustered.

\[
C(\text{Filter(TableScan(R), A\leq 100)}) = \text{NPag}(R) = 10.000
\]

\[
C(\text{IndexFilter(R, \text{Idx.R.A, A\leq 100})})
\]

\[
= \text{CI} + \text{CD} = s_f(R. A \leq 100) * \text{NLeaf(Idx.R.A)} + s_f(R. A \leq 100) * \text{NPag}(R) = 1/10 * 1.000 + 1/10 * 10.000
\]

\[
= 1.100
\]
C(Filter(TableScan(T), D≤10)) = NPag(T) = 50.000  
C(IndexFilter(T, Idx.T.D, D≤10))  
= CI + CD  
= \left[ s_{T.D \leq 10}\cdot NLeaf(Idx.T.D) \right] + \left[ s_{T.D \leq 10}\cdot NKey(T.D) \cdot \Phi(NReg(T)/NKey(T.D),NPag(T)) \right]  
= 10/1.000\cdot 20.000 + 10/1.000\cdot 100\cdot \min(10.000.000/100, 50.000) = 200 + 50.000 = 50.200  

C(Filter(TableScan(T), D=10)) = NPag(T) = 50.000  
C(IndexFilter(T, Idx.T.D, D=10))  
= CI + CD  
= \left[ s_{T.D = 10}\cdot NLeaf(Idx.T.D) \right] + \left[ \Phi(NReg(T)/NKey(T.D),NPag(T)) \right]  
= 1/100\cdot 20.000 + \min(10.000.000/100, 50.000) = 200 + 50.000 = 50.200  

d) Compute the cost of a MergeJoin plan that uses the cheapest plans for the selections (ignore the fact that IdS is not a key for neither table) (draw the plan first)  

After the condition T.D = 10 we can forget the T.D field – this is just a small optimization.  

C(IndexFilter(R, Idx.R.A, A≤100)) = 1.100 (see the previous point)  
EReg(IndexFilter(R, Idx.R.A, A≤100)) = s_{R. A \leq 100}\cdot NReg(R) = 1/10\cdot 500.000 = 50.000  
EPag(Project(IndexFilter(…), {R.IdS})) = 50.000\cdot 4/4.000 = 50  
C(Sort({R.IdS})) = C(IndexFilter) = 1.100  

C(TableScan(T)) = 50.000  
EReg(Filter(T.D=10)) = s_{T.D=10}\cdot NReg(T) = 1/100\cdot 10.000.000 = 100.000
\[ \text{NPag(} \text{Project(} \text{Filter(} \ldots \text{,} \{T.\text{IdS, T.C}\} )\text{)} = 100.000 \times 8/4.000 = 200 \]

\[ \text{C(} \text{Sort(} T \text{)} \text{)} = 2 \times \text{NPag(} \text{Project(} \text{Filter(} \ldots \text{,} \{T.\text{IdS, T.C}\} )\text{)} + \text{C(TableScan(} T \text{))} = 400 + 50.000 = 50.400 \]

\[ \text{C(MergeJoin)} = \text{C(} \text{Sort}\ldots\text{)}+\text{C(} \text{Sort}\ldots\text{)} = 1,100 + 50.400 = 51.500 \]

\[ \text{NReg(MergeJoin)} = \text{NReg(} R \text{)} \times \text{NReg(} T \text{)} \times s_\text{f}(T.\text{IdS} = T.\text{IdS}) \times s_\text{f}(R. \text{ A} \leq 100) \times s_\text{f}(T.\text{D} = 10) = \frac{500.000 \times 10.000.000}{(100.000 \times 10 \times 100)} = 50.000 \]

\[ \text{NPag(} \text{Project(} \text{MergeJoin,} \{T.C\} )\text{)} = 50.000 \times 4/4.000 = 50 \]

\[ \text{C(} \text{Sort(} \text{Project(} \text{MergeJoin,} \{T.C\} )\text{)}\text{)} = \text{C(MergeJoin)} = 51.500 \]

e) Compute the cost for an IndexNestedLoop plan where \( R \) is the outer (left-hand-side) leaf.

\[
\begin{align*}
\text{GroupBy(} \{T.C\}, \{ \text{count(*)}\} ) \quad \text{Sort(} \{T.C\} ) \\
\text{IndexNestedLoop(} R.\text{IdS} = T.\text{IdS} ) \\
\text{IndexFilter(} R, \text{IdxRA, R.A<100} ) \quad \text{Filter(} T.\text{D}=10 \text{)} \\
\text{Project(} \{R.\text{IdS}\} ) \quad \text{IndexFilter(} T, \text{IdxTIdS, R.\IdS=T.\IdS} )
\end{align*}
\]

\[ \text{C(} \text{IndexFilter(} R, \text{Idx.R.A, A\leq100)} \text{)} = 1.100 \]

\[ \text{EReg(} \text{IndexFilter(} R, \text{Idx.R.A, A\leq100)} \text{)} = 50.000 \ (\text{see above}) \]

\[ \text{C(} \text{IndexFilter(} T, \ldots, R.\text{IdS=T.\IdS}) \text{)} = CI + CD = s_\text{f}(T.\text{IdS} = T.\text{IdS}) \times \text{NLeaf(} \text{IdxTIdS} \text{)} + \Phi(\text{NReg(} T/\text{NKey(} T.\text{IdS}) , \text{NPag(} T) \text{)} \]

\[ = \left[\frac{20.000}{100.000}\right] + \Phi(10.000.000/100.000, 20.000) \]

\[ = 1 + \min(100, 20.000) = 101 \]

\[ \text{C(} \text{IndexNestedLoop)} \]

\[ = C(\text{OE}) + EReg(\text{OE}) \times C(\text{OI}) = 1.100 + 50.000 \times 101 = 1.100 + 5.050.000 = 5.051.100 \]

2. a) Assume two relation \( R(a,b) \) and \( S(a,c) \). Define the left natural outer join(\( R,S \)) (left\( \text{join}(R,S) \)) and give a very simple example.
leftjoin(R,S) contains all the tuple of join(R,S) plus all tuples t of R for which no tuple s exists in S with t.a=s.a, each extended with a null value in the c field.

Example:

R

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>Z</td>
</tr>
</tbody>
</table>

S

<table>
<thead>
<tr>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
</tbody>
</table>

LeftJoin(R,S)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>Y</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>Y</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>Z</td>
<td>null</td>
</tr>
</tbody>
</table>

b) Write a two-lines (more or less) pseudo code specification for IndexNestedLoop. Briefly specify (using natural language or pseudo-code) how one could generalize IndexNestedLoop to LeftIndexNestedLoop in order to compute left outer join

Index nested loop(R,S):

for r in R do for s in (getIndex(S,IdxS, r.a=s.a) ) do return(r+s);

LeftIndexNestedLoop(R,S):

for r in R do
    let Set = (getIndex(S,IdxSa, r.a=s.a) )
    if Set = empty then return({a=r.a, b=r.b, c=null})
    else for s in Set do return(r+s);

c) Compare the cost of LeftIndexNestedLoop(OE,OI,condition)) with the cost of IndexNestedLoop(OE,OI),condition)

LeftIndexNestedLoop(OE,OI,condition)) only adds some main-memory operations to IndexNestedLoop(OE,OI),condition), hence they have the same cost

3. a) Explain (briefly) why heap organization is the one that is most commonly used
   b) In which situations the heap organization is the best one?
   c) In which situations is the hash (procedural) organization the best one?