Homework
Query Processing

Problem 1 *Compute.*

1. Let $M = 1000$, $B(R) = 5000$, $B(S) = 2000$. Determine the costs of
   (a) block nested-loops join $R \bowtie S$;
   (b) sort-based join $R \bowtie S$;
   (c) hash-based bag $R - S$.
2. Let $M = 200$, $B(R) = 5000$. Query processor determines that to compute $R \bowtie S$ using sort-based join algorithm it needs three passes (cannot do it in two passes). What is the smallest possible size of S (in terms of number of blocks)?
3. Let $B(R) = 700$. How many buffer slots (pages) do we need in order to compute $\gamma_L(R)$ using a two-pass hash-based algorithm?
4. Let $M = 20$, $B(R) = 40,000, B(S) = 5000$. How many passes will a sort-based algorithm for $B(R) \cap B(S)$ require?

Problem 2 *Consider the following database schema describing a league of basketball teams:*

```
CREATE TABLE Teams (
   Id     INT PRIMARY KEY,
   Name   CHAR(30),
   Coach  INT REFERENCES Coaches,
   Wins   INT,
   Losses INT,
   Place  INT
 );

CREATE TABLE Players (
   Id       INT PRIMARY KEY,
   Name     CHAR(30),
   Position CHAR(2),
   Height   INT,   /* in inches */
   Team     INT REFERENCES Teams
);

CREATE TABLE Stats (
  Player  INT REFERENCES Players,
  Game    INT REFERENCES Games,
  PTS     INT, /* points scored */
  AST     INT, /* assists */
  RB      INT, /* rebounds */
  BLK     INT, /* blocks */
  STL     INT, /* steals */
  TO      INT, /* turnovers */
  PF      INT, /* personal fouls */
  TF      INT, /* technical/flagrant fouls */
  PRIMARY KEY (Player, Game)
)
```

```
CREATE TABLE Coaches (
   Id    INT PRIMARY KEY,
   Team  INT REFERENCES Teams,
   Name  CHAR(30)
);

CREATE TABLE Games (
   Id            INT  PRIMARY KEY,
   HomeTeam      INT  REFERENCES Teams,
   HomeTeamScore INT,
   AwayTeam      INT  REFERENCES Teams,
   AwayTeamScore INT
 );
```

Translate the following SQL queries into relational algebra expressions (or trees). Use merging via two-child select node for nested queries.

1. (*Find assist and turnover stats of point guards*)

```
SELECT p.Name, s.AST, s.TO
FROM Stats s, Players p,
WHERE p.Position = 'PG' and /* point guard */
      s.Player = p.Id;
```

2. (*Find players who achieved double doubles in* Surfers*' home wins.*)

```
SELECT p.Name, p.Position, t2.Name
FROM Stats s, Players p, Teams t1, Teams t2, Games g
WHERE s.Player = p.Id and
      s.game = g.Id and
```

```
      g.HomeTeam = t1.Id and
      g.AwayTeam = t2.Id and
      p.Team = t1.id   and   /* players playing for the home team */
      t1.name = 'Surfers' and
      p.PTS > 10 and
      p.RB > 10  and      /* double double */
      g.HomeTeamScore > g.AwayTeamScore;
```

3. (*Aggregate points, rebounds and assists of all players in the games they have not fouled out of by position played*)

```
SELECT p.Position, SUM(PTS), SUM(RB), SUM(AST)
FROM Players p, Stats s
WHERE p.Id = s.Player and s.PF < 6
GROUP BY p.Position;
```

4. (*Find all* Bisons *players who scored more than 200 points in league games).*

```
SELECT Name
FROM Players
WHERE Team IN (SELECT Id FROM Teams WHERE Name = 'Bisons') and
      Id IN (SELECT Player FROM Stats GROUP BY Player HAVING SUM(PTS) > 200);
```

5. (*For each team report the tallest player*)

```
 SELECT t.Name, p1.Name, p1.Height, p1.Position
 FROM Players p1, Teams t
 WHERE Height = (SELECT MAX(Height)
                FROM Players p2
                WHERE p1.Team = p2.Team ) and
       t.Id = p1.Team;
```

Problem 3 *"Anti-equivalences". Give examples that show that the following are not proper relational algebra query equivalences.*

1. $\pi_L(R \cup S) \equiv \pi_L(R) \cup \pi_L(S)$ (set union).
2. $\pi_L(R - S) \equiv \pi_L(R) - \pi_L(S)$ (either set or bag difference).
3. $\pi_L(\delta(R) \equiv \delta(\pi_L(R))$
4. $\delta(R \cup_{bag} S) \equiv \delta(R) \cup_{bag} \delta(S)$
5. $\delta(R -_{bag} S) \equiv \delta(R) -_{bag} \delta(S)$

Problem 4 *For the database in problem 2, let us abbreviate the table names as* `T, C, P, G` *and* `S` *(*`Teams`*,* `Coaches`*,* `Players`*,* `Games` *and* `Stats` *respectively). Let us also abbreviate* `Position` *as* `Pos` *and* `HomeTeam, HomeTeamScore, AwayTeam, AwayTeamScore` *as* `HT`*,* `HTS`*,* `AT` *and* `ATS`*. (this is done to keep expression length manageable).*

1. For each query from Problem 2, transform the query plan (relational algebra expression) you obtained there to a new query plan using the following set of rules:
   (a) Replace all cartesian products with appropriate joins.
   (b) Perform join operations from left-to-right, one-by-one (i.e., $(((R \bowtie S) \bowtie T) \bowtie W)$ is preferred over $((R \bowtie S) \bowtie (T \bowtie W)))$. Order relations in the join operations to achieve this.
   (c) Push selections as far down inside joins and other binary operations as possible.
   (d) Push projections as far down inside joins and other binary operations as possible, but do not push them past selections. Do NOT create new projection operations.
   (e) Do NOT push grouping operations inside joins.
2. Using the same rule-based optimizer as above, produce final query plans for the following relational algebra expressions:
   (a) $\pi_{P.Name,S.PTS}(\sigma_{S.AST>S.PTS \wedge P.Pos='PG'}(\sigma_{S.Player=P.Id}(S \times P)))$
   (b) $\pi_{P.Name}(\sigma_{S.RB>10 \wedge P.Height<70}(\sigma_{S.Player=P.Id}(S \times P))) - \pi_{P.Name}(\sigma_{T.Name='Waves'}(T \bowtie_{T.Id=P.Team} P))$
   (c) (note: ρ - renaming operation)
   $((\rho_{P1}(P) \bowtie_{P1.Team=T1.Id} (\rho_{T1}(T) \bowtie_{T1.Id=G.HT} G)) \bowtie_{G.AT=T2.Id} (\rho_{P2}(P) \bowtie_{P2.Team=T2.Id} \rho_{T2}(T)))$
   (d) $\pi_{T1.Name,T2.Name,P1.Name,P2.Name}(\sigma_{P1.Pos=P2.Pos}$ $(\sigma_{T1.Name='Dinos' \wedge T2.Name='Waves'}(((((\rho_{P1}(P) \bowtie_{P1.Team=T1.Id} (\rho_{T1}(T) \bowtie_{T1.Id=G.HT} G)) \bowtie_{G.AT=T2.Id} (\rho_{P2}(P) \bowtie_{P2.Team=T2.Id} \rho_{T2}(T))))))$