Nathan Tung
004059195

CS143 Homework #3

1. a. CHECK (weight < = 5)

   b.
```
CREATE TRIGGER LaptopWeight
AFTER INSERT on Laptop
REFERENCING NEW ROW AS NEWLT
FOR EACH ROW
WHEN NEWLT.weight > 5
BEGIN
      UPDATE Laptop
      SET weight = NULL
      WHERE model = NEWLT.model
END
```

2. We did not look into SQL assertions, but referential integrity can be found by using foreign keys.

```
ALTER TABLE S
ADD FOREIGN KEY(a) REFERENCES R(a);
```

   Otherwise, an assertion looks something like:

```
CREATE ASSERTION SRForeignKey
CHECK ... (condition in which no no foreign key violation exists)
```

3. The tuples in the table are:
   (1,8)
   (100,0)
   (100,0)

4. a.
```
CREATE VIEW EmployeeNames AS
      SELECT ename
      FROM Employees

CREATE VIEW DeptInfo AS
      SELECT dept, AVG(salary) avgSalary
      FROM Employees
      GROUP BY dept
```

   b. Mike needs to have these permissions:
      Able to see EmployeeNames view
      Able to see DeptInfo view
      Able to delete (modify) on EmployeeNames view

```
GRANT
      SELECT ON Dept Info
      TO Mike

GRANT
      SELECT, DELETE On EmployeeNames
      TO Mike
```

   c. The secretary only has SELECT permissions on the View that holds average department averages, so Mike should not be able to see any individual earnings.

d. These updates will not successfully update the underlying tables:

```
UPDATE EmployeeNames
SET ename=NULL
WHERE ename='John';

UPDATE DeptInfo
SET avgDept=60000
WHERE dept='IT';
```

The first fails because we set a primary or otherwise unique key to NULL. The second attempts to override a calculated value.

e.
```
GRANT ALL ON Employees, EmployeeNames
TO JOE
WITH GRANT OPTION
```

Joe should not be able to see the DeptInfo view unless he is given explicit permission (even though he has all privileges on the underlying table).

f. We would have to use CASCADE to remove privileges trailing down from Joe.

```
REVOKE ALL ON Employees, EmployeeNames
FROM JOE
CASCADE
```

We remove all privileges from Joe, cascading the revocation downwards so that James and Susan can no longer see our data either. We can then give access back to Joe (since he is the boss), but don't give him grant privileges.

```
GRANT ALL ON Employees, EmployeeNames
TO JOE
```

The AllNames view that Joe made should get dropped, since we took away his SELECT privilege from the EmployeeNames view (from which AllNames was made) – at least for a short time. The StaffNames relation is still out there somewhere.