1. (6 points) Here are declarations of two relations R and S:

```sql
CREATE TABLE S(
    c INT PRIMARY KEY,
    d INT
);
CREATE TABLE R(
    a INT PRIMARY KEY,
    b INT,
    CHECK(b IN (SELECT c FROM S))
);
```

Currently R and S have tuples as shown below:

<table>
<thead>
<tr>
<th>R(a, b):</th>
<th>S(c, d):</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 4</td>
<td>2 10</td>
</tr>
<tr>
<td>1 5</td>
<td>3 11</td>
</tr>
<tr>
<td>2 4</td>
<td>4 12</td>
</tr>
<tr>
<td>3 5</td>
<td>5 13</td>
</tr>
</tbody>
</table>

For the following situations, explain what happens and why. (1 point each)

(a) insert (5, 4) into R

The tuple will be inserted into R, since both key constraint and the tuple based check are satisfied.

(b) update (0, 4) in R to (0, 3)

The tuple will be updated, since both key constraint and the tuple based check are satisfied.

(c) delete (4, 12) from S

The tuple will be deleted. It does violate the tuple based check. However, the check is enforced only when there are changes made to R.

(d) insert (1, 4) into R

The insertion will be rejected due to the violation of key constraint on R.
(e) update (0, 4) in R to (0, 0)

The update will be rejected due to the violation of tuple based check on R.

(f) update (4, 12) in S to (6, 12)

The tuple will be updated. This does violate the tuple based check on R, but the check is only enforced when there are changes to R.

2. (2 points) Let’s slightly change the declaration R in problem 1 to the following. Assume S and the content of R and S remain the same.

```
CREATE TABLE R(
    a INT PRIMARY KEY,
    b INT REFERENCES S(c)
);
```

For the following situations, explain what happens and why. (1 point each)

(a) update (0, 4) in R to (0, 3)

The tuple will be updated. It doesn’t violate any constraint.

(b) delete (4, 12) from S

The deletion is rejected. The deletion will violate the foreign key constraint, and by default, it will be rejected.

3. (2 points) Here is a SQL standard trigger on relation R(a,b):

```
CREATE TRIGGER T
   AFTER INSERT ON R
   REFERENCING NEW ROW AS new_t
   FOR EACH ROW
   WHEN(new_t.a * new_t.b > 10)
   INSERT INTO R VALUES(new_t.a - 1, new_t.b + 1);
```

Suppose a user inserts (4, 3) into R. What are the tuples that are actually inserted into R?

(4, 3)

(3, 4)

(2, 5)