CSC343 Worksheet 7 Solution

June 23, 2020

1. a)

```
void askUserForPrice() {
    EXEC SQL BEGIN DECLARE SECTION;
        int model;
        float speed;
        int ram;
        int hd;
        float price;
        char maker;
        float targetPrice;

        float minDiff;
        int modelSol;
        float speedSol;
        char makerSol;
    EXEC SQL END DECLARE SECTION;

    EXEC SQL DECLARE execCursor CURSOR FOR
        SELECT * FROM Product NATURAL JOIN PC

    EXEC SQL OPEN execCursor;

    printf("Enter target price:");
    scanf("%f", &targetPrice);

    while(1) {
        EXEC SQL FETCH FROM execCursor INTO :model,
            :speed, :ram, :hd, :price, :maker;

        if (NO_MORE_TUPLES) break;

        if (abs(price - targetPrice) >= minDiff) {
            continue;
        }

        minDiff = abs(price - targetPrice);
        modelSol = model;
        speedSol = speed;
        makerSol = maker;
    }
```

```
40
41          EXEC SQL CLOSE execCursor;
42
43          printf("maker=%c, model=%d, speed=%.2f\n", makerSol, modelSol
    , speedSol);
44
45      }
46
```

Notes:

- EXEC SQL
  - Allows to use SQL statements within a host-language program
- The DECLARE Section
  - is used to declare shared variables
  - **Syntax:**
    EXEC SQL BEGIN DECLARE SECTION;
    ... // Variable declarations in any language
    EXEC SQL END DECLARE SECTION;

    Example:

```
1           void getStudio() {
2               EXEC SQL BEGIN DECLARE SECTION;
3                   char studioName[50], studioAddr[256]; // <- c
    variables
4                   char SQLSTATE[6];
5               EXEC SQL END DECLARE SECTION;
6
7               EXEC SQL INSERT INTO Studio(name, address)
8                       VALUES (:studioName, :studioAddr);
9           }
10
```

- Cursors
  - Is the most versatile way to connect SQL queries
  - **Syntax:**
    EXEC SQL DECLARE < cursor name > CURSOR FOR < query >

    EXEC SQL OPEN < cursor name >;
    ...
    EXEC SQL CLOSE < cursor name >;

    Example:

```
        void getStudio() {
            EXEC SQL BEGIN DECLARE SECTION;
                char studioName[50], studioAddr[256]; // <- c variables
                char SQLSTATE[6];
            EXEC SQL END DECLARE SECTION;

            EXEC SQL INSERT INTO Studio(name, address)
                        VALUES (:studioName, :studioAddr);
        }

```

Example in Python:

```
        import sqlite3
        connection = sqlite3.connect("company.db")

        cursor = connection.cursor()

        staff_data = [ ("William", "Shakespeare", "m", "1961-10-25"),
                       ("Frank", "Schiller", "m", "1955-08-17"),
                       ("Jane", "Wall", "f", "1989-03-14") ]

        for p in staff_data:
            format_str = """INSERT INTO employee (staff_number, fname, lname, gender, birth_date)
            VALUES (NULL, "{first}", "{last}", "{gender}", "{birthdate}");"""

            sql_command = format_str.format(first=p[0], last=p[1], gender=p[2], birthdate = p[3])
            cursor.execute(sql_command)

```

- Fetch Statement
  - fetch data from the result table one row at a time
  - **Syntax:**
    EXEC SQL FETCH FROM < cursor name > INTO < list of variables >

    Example:

```
        void worthRanges() {
            int i, digits, counts[15];
            EXEC SQL BEGIN DECLARE SECTION;
                int worth;
                char SQLSTATE[6];
            EXEC SQL END DECLARE SECTION;
            EXEC SQL DECLARE execCursor CURSOR FOR
                SELECT netWorth FROM MovieExec;
```

```
9
10              EXEC SQL OPEN execCursor;
11              for (i=1; i < 15; i++) counts[i] = 0;
12              while(1) {
13                  EXEC SQL FETCH FROM execCursor INTO :worth; //
      fetches a row of value from movieExec and stores in worth
14                  if (NO_MORE_TUPLES) break;
15
16                  ...
17              }
18          }
19
```

b)

```
1   void findLaptops() {
2       EXEC SQL BEGIN DECLARE SECTION;
3           int model;
4           float speed;
5           int ram;
6           int hd;
7           int screen;
8           float price;
9
10          float minSpeed;
11          int minRam;
12          int minHd;
13          float minPrice;
14      EXEC SQL END DECLARE SECTION;
15
16      EXEC SQL DECLARE execCursor CURSOR FOR
17          SELECT model, speed, ram, hd, screen, price, maker
18          FROM Product NATURAL JOIN Laptop;
19
20      EXEC SQL OPEN execCursor;
21
22      printf("Enter minimum speed:");
23      scanf("%f", &minSpeed);
24
25      printf("Enter minimum ram:");
26      scanf("%f", &minRam);
27
28      printf("Enter minimum hard-drive space:");
29      scanf("%f", &minHd);
30
31      printf("Enter minimum price:");
32      scanf("%f", &minPrice);
33
34      while(1) {
35          EXEC SQL FETCH FROM execCursor INTO :model,
36              :speed, :ram, :hd, :screen, :price, :maker;
37
38          if (NO_MORE_TUPLES) break;
39
40          if (
41              speed >= minSpeed &&
```

```
                ram >= minRam &&
                hd >= minHd &&
                screen >= minScreen
            ) {
                printf("model=%d, speed=%.2f, ram=%d, hd=%d, screen=%
d, price=%.2f, maker=%c",
                    model, speed, ram, hd, screen, price, maker);
            }
        }

        EXEC SQL CLOSE execCursor;
    }

```

c)

```
    void findLaptops() {
        EXEC SQL BEGIN DECLARE SECTION;
            int model;
            float speed;
            int ram;
            int hd;
            int screen;
            float price;

            float minSpeed;
            int minRam;
            int minHd;
            float minPrice;
        EXEC SQL END DECLARE SECTION;

        EXEC SQL DECLARE execCursor CURSOR FOR
            SELECT model, speed, ram, hd, screen, price, maker
            FROM Product NATURAL JOIN Laptop;

        EXEC SQL OPEN execCursor;

        printf("Enter minimum speed:");
        scanf("%f", &minSpeed);

        printf("Enter minimum ram:");
        scanf("%f", &minRam);

        printf("Enter minimum hard-drive space:");
        scanf("%f", &minHd);

        printf("Enter minimum price:");
        scanf("%f", &minPrice);

        while(1) {
            EXEC SQL FETCH FROM execCursor INTO :model,
                :speed, :ram, :hd, :screen, :price, :maker;

            if (NO_MORE_TUPLES) break;

            if (
```

```
                speed >= minSpeed &&
                ram >= minRam &&
                hd >= minHd &&
                screen >= minScreen
            ) {
                printf("model=%d, speed=%.2f, ram=%d, hd=%d, screen=%
d, price=%.2f, maker=%c",
                    model, speed, ram, hd, screen, price, maker);
            }
        }

        EXEC SQL CLOSE execCursor;
    }

```

d)

```
    #include <stdbool.h>
    #include <string.h>
    ...
    void printSpecifications() {
        EXEC SQL BEGIN DECLARE SECTION;
            int model;
            bool color;
            char printType[50];
            float price;

            float speed;
            int ram;
            int hd;
            int screen;

            char maker;
            int productModel;
            char productType[50];

            char targetMaker;
        EXEC SQL END DECLARE SECTION;

        EXEC SQL DECLARE execCursor CURSOR FOR
            SELECT DISTINCT maker, DISTINCT productType FROM Product;

        printf("Enter manufacturer:");
        scanf("%c", &targetMaker);

        EXEC SQL OPEN execCursor;
        while (1) {
            EXEC SQL FETCH FROM execCursor INTO :maker, :productType;

            if (NO_MORE_TUPLES) break;

            if (tolower(maker) != tolower(targetMaker)) continue;

            if (strcmp(productType,'pc')) {
                EXEC SQL DECLARE pcCursor CURSOR FOR
                    SELECT speed, ram, hd, price FROM PC
```

```
40                      NATURAL JOIN Product
41                      WHERE type=productType;
42
43                  EXEC SQL OPEN pcCursor;
44                  while(1) {
45                      EXEC SQL FETCH FROM pcCursor INTO :speed,
46                          :ram, :hd, :price;
47
48                      if (NO_MORE_TUPLES) break;
49
50                      printf("model=%d, speed=%.2f, ram=%d, hd=%d,
    price=%.2f, maker=%c, type=%s",
51                      model, speed, ram, hd, screen, price, maker,
    productType);
52                  }
53                  EXEC SQL CLOSE pcCursor;
54
55              } else if (strcmp(productType, 'laptop')) {
56
57                  EXEC SQL DECLARE laptopCursor CURSOR FOR
58                      SELECT speed, ram, hd, screen, price FROM Laptop
59                      NATURAL JOIN Product
60                      WHERE type=productType;
61
62                  EXEC SQL OPEN laptopCursor;
63                  while(1) {
64                      EXEC SQL FETCH FROM laptopCursor INTO :speed,
65                          :ram, :hd, :screen, :price;
66
67                      if (NO_MORE_TUPLES) break;
68
69                      printf("model=%d, speed=%.2f, ram=%d, hd=%d,
    screen=%d, price=%.2f, maker=%c, type=%s",
70                      model, speed, ram, hd, screen, screen, price,
    maker, productType);
71                  }
72                  EXEC SQL CLOSE laptopCursor;
73
74
75              } else if (strcmp(productType, 'printer')) {
76                  EXEC SQL DECLARE printerCursor CURSOR FOR
77                      SELECT color, printType, price FROM Printer
78                      NATURAL JOIN Product
79                      WHERE type=productType;
80
81                  EXEC SQL OPEN printerCursor;
82                  while(1) {
83                      EXEC SQL FETCH FROM printerCursor INTO :color,
84                          :printType, :price;
85
86                      if (NO_MORE_TUPLES) break;
87
88                      printf("model=%d, color=%s, price=%.2f, maker=%c,
     type=%s",
```

```
                    model, color ? "true" : "false", price, maker,
  type);
                }
                EXEC SQL CLOSE printerCursor;
            }
        }
        EXEC SQL CLOSE execCursor;
    }

```

Correct Solution:

```
    #include <stdbool.h>
    #include <string.h>
    ...
    void printSpecifications() {
        EXEC SQL BEGIN DECLARE SECTION;
            int model;
            bool color;
            char printType[50];
            float price;

            float speed;
            int ram;
            int hd;
            int screen;

            char maker;
            int productModel;
            char productType[50];

            char targetMaker;
        EXEC SQL END DECLARE SECTION;

        EXEC SQL DECLARE execCursor CURSOR FOR
            SELECT maker, productType FROM Product
            GROUP BY maker, productType; // <- Correction

        printf("Enter manufacturer:");
        scanf("%c", &targetMaker);

        EXEC SQL OPEN execCursor;
        while (1) {
            EXEC SQL FETCH FROM execCursor INTO :maker, :
   productType;

            if (NO_MORE_TUPLES) break;

            if (tolower(maker) != tolower(targetMaker)) continue;

            if (strcmp(productType,'pc')) {
```

```
                EXEC SQL DECLARE pcCursor CURSOR FOR
                    SELECT speed, ram, hd, price FROM PC
                    NATURAL JOIN Product
                    WHERE type=productType;

                EXEC SQL OPEN pcCursor;
                while(1) {
                    EXEC SQL FETCH FROM pcCursor INTO :speed,
                        :ram, :hd, :price;

                    if (NO_MORE_TUPLES) break;

                    printf("model=%d, speed=%.2f, ram=%d, hd=%d,
price=%.2f, maker=%c, type=%s",
                    model, speed, ram, hd, screen, price, maker,
productType);
                }
                EXEC SQL CLOSE pcCursor;

            } else if (strcmp(productType, 'laptop')) {

                EXEC SQL DECLARE laptopCursor CURSOR FOR
                    SELECT speed, ram, hd, screen, price FROM
Laptop
                    NATURAL JOIN Product
                    WHERE type=productType;

                EXEC SQL OPEN laptopCursor;
                while(1) {
                    EXEC SQL FETCH FROM laptopCursor INTO :speed,
                        :ram, :hd, :screen, :price;

                    if (NO_MORE_TUPLES) break;

                    printf("model=%d, speed=%.2f, ram=%d, hd=%d,
screen=%d, price=%.2f, maker=%c, type=%s",
                    model, speed, ram, hd, screen, screen, price,
maker, productType);
                }
                EXEC SQL CLOSE laptopCursor;


            } else if (strcmp(productType, 'printer')) {
                EXEC SQL DECLARE printerCursor CURSOR FOR
                    SELECT color, printType, price FROM Printer
                    NATURAL JOIN Product
                    WHERE type=productType;

                EXEC SQL OPEN printerCursor;
                while(1) {
                    EXEC SQL FETCH FROM printerCursor INTO :color,
                        :printType, :price;

```

```
                    if (NO_MORE_TUPLES) break;

                    printf("model=%d, color=%s, price=%.2f, maker
 =%c, type=%s",
                    model, color ? "true" : "false", price, maker,
  type);
                }
                EXEC SQL CLOSE printerCursor;
            }
        }
        EXEC SQL CLOSE execCursor;
    }

```

e)

f)

```
#include <stdbool.h>
#include <string.h>
...
void insertNewPC() {
    EXEC SQL BEGIN DECLARE SECTION;
        int model;
        float speed;
        int ram;
        int hd;
        float price;
        char maker;

        int modelCount;
    EXEC SQL END DECLARE SECTION;

    printf("Enter manufacturer:\n");
    scanf("%c", &maker);

    printf("Enter model:\n");
    scanf("%d", &model);

    printf("Enter speed:\n");
    scanf("%f", &speed);

    printf("Enter ram:\n");
    scanf("%d", &ram);

    printf("Enter hd:\n");
    scanf("%d", &hd);

    printf("Enter price:\n");
    scanf("%f", &price);

    printf("Enter maker:\n");
    scanf("%c", &maker);

    EXEC SQL DECLARE execCursor CURSOR FOR
        SELECT COUNT(model) FROM (
```

```
                (SELECT model FROM Product WHERE model=:model)
                UNION
                (SELECT model FROM PC WHERE model=:model)
            );

        EXEC SQL OPEN execCursor;
            EXEC SQL FETCH FROM execCursor INTO :modelCount;

            if (modelCount != 0) {
                printf("Error. Model already exists in database.");
            } else {
                EXEC SQL INSERT INTO PC(model, speed, ram, hd, price)
                                VALUES(:model, :speed, :ram, :hd, :
price);

                EXEC SQL INSERT INTO Product(model, maker, type)
                                VALUES(:model, :maker, "pc")
            }


        EXEC SQL CLOSE execCursor;
    }

```

2. a)

```
    void classWithLargestPower() {
        EXEC SQL BEGIN DECLARE SECTION;
            int class;
        EXEC SQL END DECLARE SECTION;

        EXEC SQL SELECT class FROM FROM Classes
            INTO :class
            WHERE numGuns * POWER(bore, 3) >= ALL (
                SELECT numGuns * POWER(bore, 3) FROM Classes
            );

        printf("Class = %s\n", class);
    }

```

b)

```
    #include <string.h>
    ...
    void countryWithMostShipsSunk() {
        EXEC SQL BEGIN DECLARE SECTION;
            char targetBattle[255];
            char country[100];
            int count;

            char mostSunkCountry[100];
            int maxSunkCount = 0;

            char mostDamagedCountry[100];
            int maxDamagedCount = 0;

```

```
15          EXEC SQL END DECLARE SECTION;
16
17          printf("Enter name of battle:\n");
18          scanf("%s", &targetBattle);
19
20          EXEC SQL DECLARE shipsSunkCursor CURSOR FOR
21              SELECT country, COUNT(Outcomes.result) FROM Classes
22              INNER JOIN Ships ON Classes.class = Ships.class
23              INNER JOIN Outcomes ON Ships.name = Outcomes.ship
24              INNER JOIN Battles ON Battles.name = Outcome.battle
25              GROUP BY country
26              HAVING Battles.name=:targetBattle;
27                     Outcomes.result='sunk';
28
29          EXEC SQL DECLARE shipsDamagedCursor CURSOR FOR
30              SELECT country, COUNT(Outcomes.result) FROM Classes
31              INNER JOIN Ships ON Classes.class = Ships.class
32              INNER JOIN Outcomes ON Ships.name = Outcomes.ship
33              INNER JOIN Battles ON Battles.name = Outcome.battle
34              GROUP BY country
35              HAVING Battles.name=:targetBattle;
36                      Outcomes.result='damaged';
37
38          EXEC SQL OPEN shipsSunkCursor;
39              while(1) {
40                  EXEC SQL FETCH FROM shipsSunkCursor INTO :country,
41                  :count;
42
43                  if (NO_MORE_TUPLES) break;
44
45                  if (count > maxSunkCount) {
46                      maxSunkCount = count;
47                      strcpy(mostSunkCountry, country);
48                  }
49              }
50
51              printf("Country with most sunk ships: %s",
    mostSunkCountry);
52
53          EXEC SQL CLOSE shipsSunkCursor;
54
55          EXEC SQL OPEN shipsDamagedCursor;
56              while(1) {
57                  EXEC SQL FETCH FROM shipsDamagedCursor INTO :country,
58                  :count;
59
60                  if (NO_MORE_TUPLES) break;
61
62                  if (count > maxDamagedCount) {
63                      maxDamagedCount = count;
64                      strcpy(mostDamagedCountry, country);
65                  }
66              }
67
```

```
68              printf("Country with most damaged ships: %s",
    mostDamagedCountry);
69
70          EXEC SQL CLOSE shipsDamagedCursor;
71
72      }
73
```