

# CSC343 Worksheet 8 Solution

June 24, 2020

1. a)

```
1  #include <float.h>
2
3  #include sqlcli.h
4
5  void askUserForPrice() {
6
7      float targetPrice, minDiff, speedSol, minDiff = FLT_MAX;
8      int modelSol;
9      char makerSol;
10
11      SQLHENV myEnv;
12      SQLHDBC myCon;
13      SQLHSTMT execStat;
14
15      SQLINTEGER model, modelInfo, speedInfo, ram, ramInfo, hd,
hdInfo, priceInfo, makerInfo;
16      SQLREAL speed, price;
17      SQLCHAR maker;
18
19
20      errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
21                                  SQL_NULL_HANDLE, &myEnv);
22
23      if (!errorCode1) {
24          errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
);
25      }
26
27      if (!errorCode2) {
28          errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
execStat)
29      }
30
31      if (!errorCode3) {
32          SQLPrepare(execStat,
33                    "SELECT model, speed, ram, hd, price, maker "
34                    "FROM Product NATURAL JOIN PC", SQL_NTS);
35          SQLExecute(execStat);
```

```

36         SQLBindCol(execStat, 1, SQL_INTEGER, &model, sizeof(model
), &modelInfo);
37         SQLBindCol(execStat, 2, SQL_FLOAT, &speed, sizeof(speed),
&speedInfo);
38         SQLBindCol(execStat, 3, SQL_INTEGER, &ram, sizeof(ram), &
ramInfo);
39         SQLBindCol(execStat, 4, SQL_INTEGER, &hd, sizeof(hd), &
hdInfo);
40         SQLBindCol(execStat, 5, SQL_FLOAT, &price, sizeof(price),
&priceInfo);
41         SQLBindCol(execStat, 5, SQL_CHAR, &maker, sizeof(maker),
&makerInfo);
42
43         printf("Enter target price:");
44         scanf("%f", &targetPrice);
45
46         while (SQLFetch(execStat) != SQL_NO_DATA) {
47
48             if (abs(price - targetPrice) >= minDiff) {
49                 continue;
50             }
51
52             minDiff = abs(price - targetPrice);
53             modelSol = model;
54             speedSol = speed;
55             makerSol = maker;
56         }
57
58         printf("maker=%c, model=%d, speed=%.2f\n", makerSol,
modelSol, speedSol);
59
60     }
61 }
62

```

### Notes:

- Using Call-Level Interface
    - Uses host language to connect to and access a database
    - Replaces embedded SQL
  - Standard SQL/CLI
    - Is database CLI for C
    - Included in file *sqlcli.h*
    - Creates deals with four kinds of records
1. Environment handle
    - \* Prepares one or more connections to database server
    - \* Is required
    - \* Is allocated using **SQLHENV**

- \* Is established via function **SQLAllocHandle**

```

1) #include sqlcli.h
2) SQLHENV myEnv;
3) SQLHDBC myCon; ← Is declared here :)
4) SQLHSTMT execStat;
5) SQLRETURN errorCode1, errorCode2, errorCode3;

6) errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv); ← Connection is prepared here :)
   (Hey DB, can I connect with you?)
7) if(!errorCode1) {
8)     errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);
9) if(!errorCode2)
10)     errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat); }

```

## 2. Connection handle

- \* Connects application program to database
- \* Is required
- \* Is declared after **SQLHENV**
- \* Is allocated using **SQLHDBC**
- \* Is established via function **SQLAllocHandle**

Sure you can

```

1) #include sqlcli.h
2) SQLHENV myEnv;
3) SQLHDBC myCon; ← Is declared here :)
4) SQLHSTMT execStat;
5) SQLRETURN errorCode1, errorCode2, errorCode3;

6) errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
7) if(!errorCode1) {
8)     errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon); ← Connection established here :)
   (Yay!!! Thank you database)
9) if(!errorCode2)
10)     errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat); }

```

## 3. Statements

- \* Created by application program (the user)
- \* Can be created as many as needed
- \* Holds information about a single SQL statement, including cursor
- \* Can represent different SQL statements at different times
- \* Is required
- \* Is declared after **SQLHDBC**
- \* Is allocated using **SQLHSTMT**
- \* Is sent using the function **SQLAllocHandle**

```

1) #include sqlcli.h
2) void worthRanges() {

3)     int i, digits, counts[15];
4)     SQLHENV myEnv;
5)     SQLHDBC myCon;
6)     SQLHSTMT execStat; ← Is declared here :)
7)     SQLINTEGER worth, worthInfo;

8)     SQLAllocHandle(SQL_HANDLE_ENV,
9)         SQL_NULL_HANDLE, &myEnv);
10)    SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);
11)    SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat); ← Statement pointer established here :)
12)    SQLPrepare(execStat,                                     (Hey DB, thank you so much for the connection!!
    "SELECT netWorth FROM MovieExec", SQL_NTS);              I will send you my SQL statement via execStat)
13)    SQLExecute(execStat);
14)    SQLBindCol(execStat, 1, SQL_INTEGER, &worth,
    sizeof(worth), &worthInfo);
15)    while(SQLFetch(execStat) != SQL_NO_DATA) {
16)        digits = 1;
17)        while((worth /= 10) > 0) digits++;
18)        if(digits <= 14) counts[digits]++;
19)    }
20)    for(i=0; i<15; i++)
21)        printf("digits = %d: number of execs = %d\n",
22)            i, counts[i]);
23) }

```

#### 4. Descriptions

- \* Holds information about either tuples or parameters
- \* Each statement has this information implicitly

#### • Processing Statements

- is done using **SQLPrepare** and **SQLExecute**

**SQLPrepare**(*sh*, *st*, *SQL\_NTS*) (1)

**SQLExecute**(*sh*) (2)

- *sh* is the statement handle created using **SQLHSTMT**
- *SQL\_NTS* evaluates the length of string in *st*

#### Example:

```

1    SQLPrepare(execStat, "SELECT netWorth FROM MovieExec",
2    SQL_NTS);
3    SQLExecute(execStat);

```

- the function **SQLExecDirect** combines **SQLPrepare** and **SQLExecute**

#### Example 2:

```

1    SQLExecDirect(execStat, "SELECT netWorth FROM MovieExec",
2    SQL_NTS);

```

#### • Fetching Data From

- Fetch
  - \* **Syntax:** **SQLFetch**(*sh*)

- \* Executes statement in **SQLPrepare** and **SQLExecute** and stores result to variable in **SQLBindCol**
  - \* Fetches a row per call
  - \* Returns a value of type **SQLRETURN**, indicating either success or error
- **SQLBindCol**
- \* **Syntax:** `SQLBindCol(sh, colNo, colType, pVar, varSize, varInfo)`
    - **sh**: the handle of statement (e.g. `execStat`)
    - **colNo**: the position of column in tuple we obtain
    - **colType**: the SQL data type of variable (e.g. `SQL_INTEGER`, `SQL_CHAR`)
    - **pVar**: the pointer to variable the value is placed
    - **varSize**: the length in bytes of the value in `pVar`
    - **varInfo**: a pointer to an integer used by `SQLBindCol` for additional value about the value produced
  - \* Stores data from **SQLFetch** to host-language variable
  - \* Must be setup before `SQLFetch(sh)` is run

```

1) #include sqlcli.h
2) void worthRanges() {

3)     int i, digits, counts[15];
4)     SQLHENV myEnv;
5)     SQLHDBC myCon;
6)     SQLHSTMT execStat;
7)     SQLINTEGER worth, worthInfo;

8)     SQLAllocHandle(SQL_HANDLE_ENV,
9)         SQL_NULL_HANDLE, &myEnv);
10)    SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);
11)    SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);
12)    SQLPrepare(execStat,
13)        "SELECT netWorth FROM MovieExec", SQL_NTS);
14)    SQLExecute(execStat);
15)    SQLBindCol(execStat, 1, SQL_INTEGER, &worth,
16)        sizeof(worth), &worthInfo);
17)    while(SQLFetch(execStat) != SQL_NO_DATA) {
18)        digits = 1;
19)        while((worth /= 10) > 0) digits++;
20)        if(digits <= 14) counts[digits]++;
21)    }
22)    for(i=0; i<15; i++)
23)        printf("digits = %d: number of execs = %d\n",
24)            i, counts[i]);
25) }

```

The value to fetch is defined here :)

The storage location is defined here :)  
(Hey DB, when data is fetched, could you store the fetched value of SQL\_INTEGER datatype to worth variable? Here is the address)

Value is fetched here :)

b)

```

#include sqlcli.h

void findLaptops() {

    float minSpeed, minPrice;
    int minRam, minHd;

    SQLINTEGER model, ram, hd, screen;
    SQLREAL speed, price;

    errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
        SQL_NULL_HANDLE, &myEnv);

```

```
14         if (!errorCode1) {
15             errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
16         );
17         }
18         if (!errorCode2) {
19             errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
20             execStat)
21         }
22         if (!errorCode3) {
23             EXEC SQL DECLARE execCursor CURSOR FOR
24             SELECT model, speed, ram, hd, screen, price, maker
25             FROM Product NATURAL JOIN Laptop;
26
27             printf("Enter minimum speed:");
28             scanf("%f", &minSpeed);
29
30             printf("Enter minimum ram:");
31             scanf("%f", &minRam);
32
33             printf("Enter minimum hard-drive space:");
34             scanf("%f", &minHd);
35
36             printf("Enter minimum price:");
37             scanf("%f", &minPrice);
38
39             while(1) {
40                 EXEC SQL FETCH FROM execCursor INTO :model,
41                     :speed, :ram, :hd, :screen, :price, :maker;
42
43                 if (NO_MORE_TUPLES) break;
44
45                 if (
46                     speed >= minSpeed &&
47                     ram >= minRam &&
48                     hd >= minHd &&
49                     screen >= minScreen
50                 ) {
51                     printf("model=%d, speed=%.2f, ram=%d, hd=%d,
52                     screen=%d, price=%.2f, maker=%c",
53                         model, speed, ram, hd, screen, price, maker);
54                 }
55
56                 EXEC SQL CLOSE execCursor;
57
58             }
59         }
60     }
```