

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, 6, 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, modelInfo, speedInfo, ram, ramInfo, hd,
    hdInfo, priceInfo, makerInfo, screen, screenInfo;
    SQLREAL speed, price;
    SQLCHAR maker;

    errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,

```

```

13         SQL_NULL_HANDLE, &myEnv);
14
15         if (!errorCode1) {
16             errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
17         );
18         }
19
20         if (!errorCode2) {
21             errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
22             execStat)
23         }
24
25         if (!errorCode3) {
26             SQLPrepare(execStat,
27                 "SELECT model, speed, ram, hd, screen, price,
28                 maker "
29                 "FROM Product NATURAL JOIN Laptop", SQL_NTS);
30             SQLExecute(execStat);
31             SQLBindCol(execStat, 1, SQL_INTEGER, &model, sizeof(model
32             ), &modelInfo);
33             SQLBindCol(execStat, 2, SQL_FLOAT, &speed, sizeof(speed), &
34             speedInfo);
35             SQLBindCol(execStat, 3, SQL_INTEGER, &ram, sizeof(ram), &
36             ramInfo);
37             SQLBindCol(execStat, 4, SQL_INTEGER, &hd, sizeof(hd), &
38             hdInfo);
39             SQLBindCol(execStat, 5, SQL_INTEGER, &screen, sizeof(
40             screen), &screenInfo);
41             SQLBindCol(execStat, 6, SQL_FLOAT, &price, sizeof(price),
42             &priceInfo);
43             SQLBindCol(execStat, 7, SQL_CHAR, &maker, sizeof(maker),
44             &makerInfo);
45
46             printf("Enter minimum speed:");
47             scanf("%f", &minSpeed);
48
49             printf("Enter minimum ram:");
50             scanf("%f", &minRam);
51
52             printf("Enter minimum hard-drive space:");
53             scanf("%f", &minHd);
54
55             printf("Enter minimum price:");
56             scanf("%f", &minPrice);
57
58             while(SQLFetch(execStat) != SQL_NO_DATA) {
59                 if (
60                     speed >= minSpeed &&
61                     ram >= minRam &&
62                     hd >= minHd &&
63                     screen >= minScreen
64                 ) {
65                     printf("model=%d, speed=%.2f, ram=%d, hd=%d,
66                     screen=%d, price=%.2f, maker=%c",

```

```

56         model, speed, ram, hd, screen, price, maker);
57     }
58 }
59 }
60 }
61

```

```

c) #include <stdbool.h>
2  #include <string.h>
3  ...
4  void printSpecifications() {
5      char targetMaker;
6
7      SQLHENV myEnv;
8      SQLHDBC myCon;
9      SQLHSTMT execStat;
10
11     SQLINTEGER model, modelInfo, speedInfo, ram, ramInfo, hd,
hdInfo, priceInfo, makerInfo, screen, screenInfo, color, colorInfo
;
12     SQLREAL speed, price;
13     SQLCHAR maker, printType[50];
14
15     SQLRETURN errorCode1, errorCode2, errorCode3;
16
17     errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
18                               SQL_NULL_HANDLE, &myEnv);
19
20     if (!errorCode1) {
21         errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
);
22     }
23
24     if (!errorCode2) {
25         errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
execStat)
26     }
27
28     if (!errorCode3) {
29         printf("Enter manufacturer:");
30         scanf("%c", &targetMaker);
31
32         SQLPrepare(execStat,
33                  "SELECT maker, productType FROM Product "
34                  "GROUP BY maker, productType "
35                  "WHERE maker = ?", SQL_NTS);
36         SQLBindParameter(execStat, 1, ..., targetMaker, ...);
37         SQLExecute(execStat);
38         SQLBindCol(execStat, 1, SQL_CHAR, &maker, sizeof(maker),
&makerInfo);
39         SQLBindCol(execStat, 2, SQL_CHAR, &productType, sizeof(
productType), &productTypeInfo);
40
41         while (SQLFetch(execStat) != SQL_NO_DATA) {

```

```

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

```



```
90         :printType, :price;
91
92         if (NO_MORE_TUPLES) break;
93
94         printf("model=%d, color=%s, price=%.2f, maker
95         =%c, type=%s",
96         , type);
97         EXEC SQL CLOSE printerCursor;
98     }
99 }
100 }
101 }
102 }
```