

CSC343 Worksheet 8 Solution

June 25, 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, subExecStat;
10
11     SQLINTEGER model, modelInfo, speedInfo, ram, ramInfo, hd,
hdInfo, priceInfo, makerInfo, screen, screenInfo, color, colorInfo
, printTypeInfo;
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         errorCode4 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
subExecStat);
27     }
28
29     if (!errorCode3 && !errorCode4) {
30         printf("Enter manufacturer:");
31         scanf("%c", &targetMaker);
32
33         SQLPrepare(execStat,
34                   "SELECT maker, productType FROM Product "
35                   "GROUP BY maker, productType "
36                   "WHERE maker = ?", SQL_NTS);
37         SQLBindParameter(execStat, 1, ..., targetMaker, ...);
38         SQLExecute(execStat);
39         SQLBindCol(execStat, 1, SQL_CHAR, &maker, sizeof(maker),
&makerInfo);
40         SQLBindCol(execStat, 2, SQL_CHAR, &productType, sizeof(
productType), &productTypeInfo);

```

```

41         while (SQLFetch(execStat) != SQL_NO_DATA) {
42             if (strcmp(productType, 'pc')) {
43                 SQLPrepare(subExecStat,
44                     "SELECT speed, ram, hd, price FROM PC
45
46                     \"NATURAL JOIN Product \"
47                     \"WHERE type= ?\", SQL_NTS);
48                 SQLBindParameter(subExecStat, 1, ...,
productType, ...);
49                 SQLExecute(subExecStat);
50
51                 SQLBindCol(subExecStat, 1, SQL_FLOAT, &speed,
sizeof(speed), &speedInfo);
52                 SQLBindCol(subExecStat, 2, SQL_INTEGER, &ram,
sizeof(ram), &ramInfo);
53                 SQLBindCol(subExecStat, 3, SQL_INTEGER, &hd,
sizeof(hd), &hdInfo);
54                 SQLBindCol(subExecStat, 4, SQL_FLOAT, &price,
sizeof(price), &priceInfo);
55
56                 while(SQLFetch(subExecStat) != SQL_NO_DATA) {
57                     printf("model=%d, speed=%.2f, ram=%d, hd=%d,
price=%.2f, maker=%c, type=%s",
58                         model, speed, ram, hd, screen, price, maker,
productType);
59                 }
60             } else if (strcmp(productType, 'laptop')) {
61
62                 SQLPrepare(subExecStat,
63                     "SELECT speed, ram, hd, screen, price
64
65                     \"NATURAL JOIN Product \"
66                     \"WHERE type= ?\", SQL_NTS);
67                 SQLBindParameter(subExecStat, 1, ...,
productType, ...);
68                 SQLExecute(subExecStat);
69
70                 SQLBindCol(subExecStat, 1, SQL_FLOAT, &speed,
sizeof(speed), &speedInfo);
71                 SQLBindCol(subExecStat, 2, SQL_INTEGER, &ram,
sizeof(ram), &ramInfo);
72                 SQLBindCol(subExecStat, 3, SQL_INTEGER, &hd,
sizeof(hd), &hdInfo);
73                 SQLBindCol(subExecStat, 4, SQL_INTEGER, &screen,
sizeof(screen), &screenInfo);
74                 SQLBindCol(subExecStat, 5, SQL_FLOAT, &price,
sizeof(price), &priceInfo);
75
76                 while(SQLFetch(subExecStat) != SQL_NO_DATA) {
77                     printf("model=%d, speed=%.2f, ram=%d, hd=%d,
screen=%d, price=%.2f, maker=%c, type=%s",
78                         model, speed, ram, hd, screen, screen, price,
maker, productType);

```



```

78         }
79     } else if (strcmp(productType, 'printer')) {
80         SQLPrepare(subExecStat,
81             "SELECT color, printType, price FROM
Printer "
82             "NATURAL JOIN Product "
83             "WHERE type= ?", SQL_NTS);
84         SQLBindParameter(subExecStat, 1, ...,
productType, ...);
85         SQLExecute(subExecStat);
86
87         SQLBindCol(subExecStat, 1, SQL_INTEGER, &color,
sizeof(speed), &speedInfo);
88         SQLBindCol(subExecStat, 2, SQL_CHAR, &printType,
sizeof(printType), &printTypeInfo);
89         SQLBindCol(subExecStat, 3, SQL_FLOAT, &price,
sizeof(price), &priceInfo);
90
91         while(SQLFetch(subExecStat) != SQL_NO_DATA) {
92             printf("model=%d, color=%s, price=%.2f, maker
=%c, type=%s",
93                 model, color ? "true" : "false", price, maker
, type);
94         }
95     }
96 }
97 }
98 }
99

```

d)

```

e) #include <sqlcli.h>
2  #include <string.h>
3  ...
4  void insertNewPC() {
5
6      int model, ram, hd;
7      float speed, price;
8      char maker;
9
10     SQLINTEGER modelCount;
11
12     SQLHENV myEnv;
13     SQLHDBC myCon;
14     SQLHSTMT execStat, subExecStat;
15
16     SQLRETURN errorCode1, errorCode2, errorCode3;
17
18     errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
19         SQL_NULL_HANDLE, &myEnv);
20
21     if (!errorCode1) {
22         errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
);

```

```

23     }
24
25     if (!errorCode2) {
26         errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
execStat);
27     }
28
29     if (!errorCode3) {
30         printf("Enter manufacturer:\n");
31         scanf("%c", &maker);
32
33         printf("Enter model:\n");
34         scanf("%d", &model);
35
36         printf("Enter speed:\n");
37         scanf("%f", &speed);
38
39         printf("Enter ram:\n");
40         scanf("%d", &ram);
41
42         printf("Enter hd:\n");
43         scanf("%d", &hd);
44
45         printf("Enter price:\n");
46         scanf("%f", &price);
47
48         printf("Enter maker:\n");
49         scanf("%c", &maker);
50
51         SQLPrepare(execStat,
52                     "SELECT COUNT(model) FROM ("
53                     "(SELECT model FROM Product WHERE model=:model)
54
55                     "UNION "
56                     "(SELECT model FROM PC WHERE model= ?)",
SQL_NTS);
57         SQLBindParameter(execStat, 1, ..., model, ...);
58         SQLExecute(execStat);
59         SQLBindCol(execStat, 1, SQL_INT, &modelCount, sizeof(
modelCount), &modelCountInfo);
60
61         if (modelCount != 0) {
62             printf("Error. Model already exists in database.");
63         } else {
64             SQLPrepare(execStat,
65                         "INSERT INTO PC(model, speed, ram, hd, price)
66
67                         "VALUES(?, ?, ?, ?, ?)", SQL_NTS);
68             SQLBindParameter(execStat, 1, ..., model, ...);
69             SQLBindParameter(execStat, 2, ..., speed, ...);
70             SQLBindParameter(execStat, 3, ..., ram, ...);
71             SQLBindParameter(execStat, 4, ..., hd, ...);
72             SQLBindParameter(execStat, 5, ..., price, ...);
73             SQLExecute(execStat);

```

```

72         SQLPrepare(execStat,
73                     "INSERT INTO Product(model, maker, type)"
74                     "VALUES(?, ?, 'pc')", SQL_NTS);
75         SQLBindParameter(execStat, 1, ..., model, ...);
76         SQLBindParameter(execStat, 2, ..., maker, ...);
77         SQLExecute(execStat);
78     }
79 }
80 }
81 }
82

```

2. a)

```

void classWithLargestPower() {
2
3     SQLINTEGER classInfo;
4     SQLCHAR class[100];
5
6     SQLHENV myEnv;
7     SQLHDBC myCon;
8     SQLHSTMT execStat, subExecStat;
9
10    SQLRETURN errorCode1, errorCode2, errorCode3;
11
12    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 class FROM FROM Classes"
28                    "WHERE numGuns * POWER(bore, 3) >= ALL ( "
29                    "SELECT numGuns * POWER(bore, 3) FROM Classes "
30                    ")", SQL_NTS);
31        SQLBindParameter(execStat, 1, ..., model, ...);
32        SQLExecute(execStat);
33        SQLBindCol(execStat, 1, SQL_CHAR, &class, sizeof(class),
34    &classInfo);
35
36        while(SQLFetch(execStat) != SQL_NO_DATA) {
37            printf("Class = %s\n", class);
38        }
39    }
40 }

```

```

b)  #include <sqlcli.h>
    2  #include <string.h>
    3  ...
    4  void countryWithMostShipsSunk() {
    5      char targetBattle[255];
    6      char mostSunkCountry[100];
    7      int maxSunkCount = 0, loopIndex = 0;
    8
    9      char mostDamagedCountry[100];
   10      int maxDamagedCount = 0;
   11
   12      SQLCHAR country[100];
   13      SQLINTEGER count, countInfo. countryInfo;
   14
   15      SQLHENV myEnv;
   16      SQLHDBC myCon;
   17      SQLHSTMT execStat, subExecStat;
   18
   19      SQLRETURN errorCode1, errorCode2, errorCode3;
   20
   21      errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
   22                                SQL_NULL_HANDLE, &myEnv);
   23
   24      if (!errorCode1) {
   25          errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
   26      );
   27      }
   28
   29      if (!errorCode2) {
   30          errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
   31          execStat)
   32      }
   33
   34      if (!errorCode3) {
   35          printf("Enter name of battle:\n");
   36          scanf("%s", &targetBattle);
   37
   38          SQLPrepare(execStat,
   39                  "SELECT country, COUNT(Outcomes.result) FROM
   40                  Classes "
   41                  "INNER JOIN Ships ON Classes.class = Ships.
   42                  class "
   43                  "INNER JOIN Outcomes ON Ships.name = Outcomes
   44                  .ship "
   45                  "INNER JOIN Battles ON Battles.name = Outcome
   46                  .battle "
   47                  "GROUP BY country "
   48                  "HAVING Battles.name=:targetBattle AND"
   49                  "Outcomes.result='sunk'", SQL_NTS);
   50          SQLExecute(execStat);
   51          SQLBindCol(execStat, 1, SQL_CHAR, &country, sizeof(
   52          country), &countryInfo);
   53          SQLBindCol(execStat, 2, SQL_INTEGER, &count, sizeof(count

```

```

47     ), &countInfo);
48
49     while(SQLFetch(execStat) != SQL_NO_DATA) {
50         if (loopIndex == 0) {
51             strcpy(mostSunkCountry, country);
52         }
53
54         if (count > maxSunkCount) {
55             maxSunkCount = count;
56             strcpy(mostSunkCountry, country);
57         }
58         loopIndex = loopIndex + 1;
59     }
60
61     printf("Country with most sunk ships: %s",
62     mostSunkCountry);
63
64     count = 0;
65     loopIndex = 0;
66     SQLPrepare(execStat,
67     Classes "
68         "INNER JOIN Ships ON Classes.class = Ships.
69     class "
70         "INNER JOIN Outcomes ON Ships.name = Outcomes
71     .ship "
72         "INNER JOIN Battles ON Battles.name = Outcome
73     .battle "
74         "GROUP BY country "
75         "HAVING Battles.name=:targetBattle AND"
76         "Outcomes.result='damaged'", SQL_NTS);
77     SQLExecute(execStat);
78
79     while(SQLFetch(execStat) != SQL_NO_DATA) {
80         if (loopIndex == 0) {
81             strcpy(mostDamagedCountry, country);
82         }
83
84         if (count > maxDamagedCount) {
85             maxDamagedCount = count;
86             strcpy(mostDamagedCountry, country);
87         }
88         loopIndex = loopIndex + 1;
89     }
90
91     printf("Country with most damaged ships: %s",
92     mostDamagedCountry);
93 }
94 }

```

```

c) #include <sqlcli.h>
2

```

```

3  void insertClassAndShip() {
4
5      char class[100], type[2], country[100], shipName[100],
6      dateLaunched[11];
7      int numGuns, bore, displacement;
8
9      SQLHENV myEnv;
10     SQLHDBC myCon;
11     SQLHSTMT execStat, subExecStat;
12
13     SQLRETURN errorCode1, errorCode2, errorCode3;
14
15     errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
16                                SQL_NULL_HANDLE, &myEnv);
17
18     if (!errorCode1) {
19         errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
20 );
21     }
22
23     if (!errorCode2) {
24         errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
25 execStat)
26     }
27
28     if (!errorCode3) {
29         printf("Enter name of class:\n");
30         scanf("%s", class);
31
32         printf("Enter name of type ('bb' or 'bc'):\n");
33         scanf("%s", type);
34
35         printf("Enter name of country:\n");
36         scanf("%s", country);
37
38         printf("Enter name of numGuns:\n");
39         scanf("%d", &numGuns);
40
41         printf("Enter name of bore:\n");
42         scanf("%d", &bore);
43
44         printf("Enter name of displacement:\n");
45         scanf("%d", &displacement);
46
47         printf("Enter name of ship (if first ship, skip by
48 pressing ENTER):\n");
49         fgets(shipName, sizeof shipName, stdin);
50
51         if (shipName[0] == '\n') {
52             strncpy(shipName, class, sizeof(class));
53         }
54
55         printf("Enter date launched (YYYY-MM-DD):\n");

```

```

53         scanf("%s", dateLaunched);
54
55         SQLPrepare(execStat,
56             "INSERT INTO Classes(class, type, country,
numGuns, bore, displacement)"
57             "VALUES (?, ?, ?, ?, ?, ?)", SQL_NTS);
58         SQLBindParameter(execStat, 1, ..., class, ...);
59         SQLBindParameter(execStat, 2, ..., type, ...);
60         SQLBindParameter(execStat, 3, ..., country, ...);
61         SQLBindParameter(execStat, 4, ..., numGuns, ...);
62         SQLBindParameter(execStat, 5, ..., bore, ...);
63         SQLBindParameter(execStat, 6, ..., displacement, ...)
64     ;
65
66     SQLExecute(execStat);
67
68     SQLPrepare(execStat,
69         "INSERT INTO Ships(name, class, launched)"
70         "VALUES (?, ?, ?)", SQL_NTS);
71     SQLBindParameter(execStat, 1, ..., shipName, ...);
72     SQLBindParameter(execStat, 2, ..., class, ...);
73     SQLBindParameter(execStat, 3, ..., dateLaunched, ...)
74 ;
75     SQLExecute(execStat);
76 }
77 }

```

d)

```

1  #include <sqlcli.h>
2
3  void correctError() {
4      SQLCHAR battle[101], shipName[101];
5      DATE_STRUCT dateLaunched;
6      DATE_STRUCT dateBattle;
7
8      int newDateLaunchedDay;
9      int newDateLaunchedMonth;
10     int newDateLaunchedYear;
11
12     int newDateBattleDay;
13     int newDateBattleMonth;
14     int newDateBattleYear;
15
16     SQLCHAR name[101], class[101]
17
18     SQLHENV myEnv;
19     SQLHDBC myCon;
20     SQLHSTMT execStat, subExecStat;
21
22     SQLRETURN errorCode1, errorCode2, errorCode3;
23
24     errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
25         SQL_NULL_HANDLE, &myEnv);
26
27     if (!errorCode1) {

```

```

28         errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon
29     );
30     }
31     if (!errorCode2) {
32         errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
33     execStat)
34     }
35     if (!errorCode3) {
36
37         SQLPrepare(execStat,
38     "SELECT Ships.name, Ships.class, Outcomes.
39     battle"
40     "FROM Ships "
41     "INNER JOIN Outcomes ON Ships.name = Outcomes
42     .ship "
43     "INNER JOIN Battles ON Outcomes.battle =
44     Battles.name "
45     "WHERE Ships.launches > Battles.date",
46     SQL_NTS);
47     SQLExecute(execStat);
48     SQLBindCol(execStat, 1, SQL_CHAR, &shipName, sizeof(
49     shipName), &shipNameInfo);
50     SQLBindCol(execStat, 2, SQL_FLOAT, &class, sizeof(class),
51     &classInfo);
52     SQLBindCol(execStat, 3, SQL_INTEGER, &battle, sizeof(
53     battle), &battleInfo);
54
55     while(SQLFetch(execStat) != SQL_NO_DATA) {
56
57         printf("Error. Ship %s is launched after date of
58     battle.\n");
59
60         printf("Enter correct launched date (YYYY-MM-DD,
61     Press enter to skip):\n");
62         fgets(dateLaunched, sizeof(dateLaunched), stdin);
63
64         if (dateLaunched[0] != '\n') {
65             // Correct date of launch
66             EXEC SQL UPDATE Ships
67                 SET launched = newDateLaunched
68                 WHERE name=:shipName AND
69                 class=:class AND
70                 launched=:dateLaunched;
71         }
72
73         printf("Enter correct battle date (YYYY-MM-DD, Press
74     enter to skip):\n");
75         fgets(dateBattle, sizeof(dateBattle), stdin);
76
77         if (dateBattle[0] != '\n') {
78             // Correct date of battle
79             EXEC SQL UPDATE Battles

```



```
70      SET date = newDateBattle
71      WHERE name=:battle AND
72            date=dateBattle;
73    }
74  }
75 }
76
77
```