CSC343 Worksheet 8 Solution

June 26, 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,   ← Connection is prepared here :)
         SQL_NULL_HANDLE, &myEnv);                    (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

   ∗ Conenects application program to database

   ∗ Is required

   ∗ Is declared after **SQLHENV**

   ∗ Is allocated using **SQLHDBC**

   ∗ 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;

Sure you can
6)   errorCode1 = SQLAllocHandle(SQL_HANDLE_ENV,
         SQL_NULL_HANDLE, &myEnv);
7)   if(!errorCode1) {
8)       errorCode2 = SQLAllocHandle(SQL_HANDLE_DBC,   ← Connection established here :)
             myEnv, &myCon);                              (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,
                SQL_NULL_HANDLE, &myEnv);
9)       SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);
10)      SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);   ←—— Statement pointer established here :)
11)      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)
12)      SQLExecute(execStat);
13)      SQLBindCol(execStat, 1, SQL_INTEGER, &worth,
                sizeof(worth), &worthInfo);                      (Hehe. Here it comes XD. Thank you DB!!)
14)      while(SQLFetch(execStat) != SQL_NO_DATA) {
15)          digits = 1;
16)          while((worth /= 10) > 0) digits++;
17)          if(digits <= 14) counts[digits]++;
         }
18)      for(i=0; i<15; i++)
19)          printf("digits = %d: number of execs = %d\n",
                    i, counts[i]);
     }
```

4. Descriptions
   * Holds information about either tuples or parameters
   * Each statement has this information implicitly

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

$$\textbf{SQLPrepare}(sh, st, SQL\_NTS) \tag{1}$$
$$\textbf{SQLExecute}(sh) \tag{2}$$

  - 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" ,
     SQL_NTS );
2    SQLExecute ( execStat );
3
```

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

    Example 2:

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

- Fetching Data From
  - Fetch
    * **Syntax:** SQLFetch(sh)

4

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

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)
```
1    #include sqlcli.h
2
3    void findLaptops() {
4
5        float minSpeed, minPrice;
6        int minRam, minHd;
7
8        SQLINTEGER model, modelInfo, speedInfo, ram, ramInfo, hd,
     hdInfo, priceInfo, makerInfo, screen, screenInfo;
9        SQLREAL speed, price;
10       SQLCHAR maker;
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        if (!errorCode2) {
20              errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
     execStat)
21        }

22

23        if (!errorCode3) {
24              SQLPrepare(execStat,
25                         "SELECT model, speed, ram, hd, screen, price,
     maker "
26                         "FROM Product NATURAL JOIN Laptop", SQL_NTS);
27              SQLExecute(execStat);
28              SQLBindCol(execStat, 1, SQL_INTEGER, &model, sizeof(model
     ), &modelInfo);
29              SQLBindCol(execStat, 2, SQL_FLOAT, &speed, sizeof(speed),
      &speedInfo);
30              SQLBindCol(execStat, 3, SQL_INTEGER, &ram, sizeof(ram), &
     ramInfo);
31              SQLBindCol(execStat, 4, SQL_INTEGER, &hd, sizeof(hd), &
     hdInfo);
32              SQLBindCol(execStat, 5, SQL_INTEGER, &screen, sizeof(
     screen), &screenInfo);
33              SQLBindCol(execStat, 6, SQL_FLOAT, &price, sizeof(price),
      &priceInfo);
34              SQLBindCol(execStat, 7, SQL_CHAR, &maker, sizeof(maker),
     &makerInfo);

35

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

38

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

41

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

44

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

47

48              while(SQLFetch(execStat) != SQL_NO_DATA) {
49                  if (
50                      speed >= minSpeed &&
51                      ram >= minRam &&
52                      hd >= minHd &&
53                      screen >= minScreen
54                  ) {
55                      printf("model=%d, speed=%.2f, ram=%d, hd=%d,
     screen=%d, price=%.2f, maker=%c",
```

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

c)
```
 1      #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              SQLBindCol ( execStat , 1, SQL_CHAR , &maker , sizeof ( maker ),
    & makerInfo );
34              SQLBindCol ( execStat , 2, SQL_CHAR , & productType , sizeof (
    productType ), & productTypeInfo );
35
36              while ( SQLFetch ( execStat ) != SQL_NO_DATA ) {
37                  if ( strcmp ( productType ,'pc ')) {
38                      SQLPrepare ( subExecStat ,
39                              " SELECT speed , ram , hd , price FROM PC
    "
```

7

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

```
77                             "WHERE type= ?", SQL_NTS);
78                     SQLBindParameter(subExecStat, 1, ...,
     productType, ...);
79                  SQLExecute(subExecStat);
80
81                  SQLBindCol(subExecStat, 1, SQL_INTEGER, &color,
     sizeof(speed), &speedInfo);
82                  SQLBindCol(subExecStat, 2, SQL_CHAR, &printType,
     sizeof(printType), &printTypeInfo);
83                  SQLBindCol(subExecStat, 3, SQL_FLOAT, &price,
     sizeof(price), &priceInfo);
84
85                  while(SQLFetch(subExecStat) != SQL_NO_DATA) {
86                      printf("model=%d, color=%s, price=%.2f, maker
     =%c, type=%s",
87                      model, color ? "true" : "false", price, maker
     , type);
88                  }
89              }
90          }
91      }
92   }
93
```

d)

e)
```
1    #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                    "UNION "
55                    "(SELECT model FROM PC WHERE model= ?)",
    SQL_NTS);
56                SQLBindParameter(execStat, 1, ..., model, ...);
57            SQLExecute(execStat);
58            SQLBindCol(execStat, 1, SQL_INT, &modelCount, sizeof(
    modelCount), &modelCountInfo);
59
60            if (modelCount != 0) {
61                printf("Error. Model already exists in database.");
62            } else {
63                SQLPrepare(execStat,
64                        "INSERT INTO PC(model, speed, ram, hd, price)
    "
65                        "VALUES(?, ?, ?, ?, ?)", SQL_NTS);
66                    SQLBindParameter(execStat, 1, ..., model, ...);
67                    SQLBindParameter(execStat, 2, ..., speed, ...);
68                    SQLBindParameter(execStat, 3, ..., ram, ...);
69                    SQLBindParameter(execStat, 4, ..., hd, ...);
70                    SQLBindParameter(execStat, 5, ..., price, ...);
71                SQLExecute(execStat);
72
73                SQLPrepare(execStat,
74                        "INSERT INTO Product(model, maker, type)"
75                        "VALUES(?, ?, 'pc')", SQL_NTS);
76                    SQLBindParameter(execStat, 1, ..., model, ...);
77                    SQLBindParameter(execStat, 2, ..., maker, ...);
```

```
78                    SQLExecute ( execStat );
79                }
80            }
81        }
82
```

2. a)
```
1      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         if (!errorCode2) {
20             errorCode3 = SQLAllocHandle ( SQL_HANDLE_STMT , myCon , &
   execStat );
21         }
22
23         if (!errorCode3) {
24             SQLPrepare ( execStat ,
25                     "SELECT class FROM FROM Classes"
26                     "WHERE numGuns * POWER(bore , 3) >= ALL ( "
27                     "SELECT numGuns * POWER(bore , 3) FROM Classes "
28                     ")", SQL_NTS );
29             SQLBindParameter ( execStat , 1, ..., model , ...);
30             SQLExecute ( execStat );
31             SQLBindCol ( execStat , 1, SQL_CHAR , &class , sizeof(class),
   &classInfo );
32
33             while( SQLFetch ( execStat ) != SQL_NO_DATA ) {
34                 printf("Class = %s\n", class );
35             }
36         }
37     }
38
```

b)
```
1      #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       if (!errorCode2) {
29           errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
   execStat)
30       }
31
32       if (!errorCode3) {
33
34           printf("Enter name of battle:\n");
35           scanf("%s", &targetBattle);
36
37           SQLPrepare(execStat,
38                       "SELECT country, COUNT(Outcomes.result) FROM
   Classes "
39                       "INNER JOIN Ships ON Classes.class = Ships.
   class "
40                       "INNER JOIN Outcomes ON Ships.name = Outcomes
   .ship "
41                       "INNER JOIN Battles ON Battles.name = Outcome
   .battle "
42                       "GROUP BY country "
43                       "HAVING Battles.name=:targetBattle AND"
44                       "Outcomes.result='sunk'", SQL_NTS);
45           SQLExecute(execStat);
46           SQLBindCol(execStat, 1, SQL_CHAR, &country, sizeof(
   country), &countryInfo);
47           SQLBindCol(execStat, 2, SQL_INTEGER, &count, sizeof(count
   ), &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",
     mostSunkCountry);
62
63
64               count = 0;
65               loopIndex = 0;
66               SQLPrepare(execStat,
67                         "SELECT country, COUNT(Outcomes.result) FROM
     Classes "
68                         "INNER JOIN Ships ON Classes.class = Ships.
     class "
69                         "INNER JOIN Outcomes ON Ships.name = Outcomes
     .ship "
70                         "INNER JOIN Battles ON Battles.name = Outcome
     .battle "
71                         "GROUP BY country "
72                         "HAVING Battles.name=:targetBattle AND"
73                         "Outcomes.result='damaged'", SQL_NTS);
74               SQLExecute(execStat);
75
76               while(SQLFetch(execStat) != SQL_NO_DATA) {
77                    if (loopIndex == 0) {
78                         strcpy(mostDamagedCountry, country);
79                    }
80
81                    if (count > maxDamagedCount) {
82                         maxDamagedCount = count;
83                         strcpy(mostDamagedCountry, country);
84                    }
85                    loopIndex = loopIndex + 1;
86               }
87
88               printf("Country with most damaged ships: %s",
     mostDamagedCountry);
89          }
90      }
91


c)     #include <sqlcli.h>
2
3      void insertClassAndShip() {
4
5          char class[100], type[2], country[100], shipName[100],
     dateLaunched[11];
6          int numGuns, bore, displacement;
```

```
7
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         if (!errorCode2) {
23             errorCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
     execStat)
24         }
25
26         if (!errorCode3) {
27             printf("Enter name of class:\n");
28             scanf("%s", class);
29
30             printf("Enter name of type ('bb' or 'bc'):\n");
31             scanf("%s", type);
32
33             printf("Enter name of country:\n");
34             scanf("%s", country);
35
36             printf("Enter name of numGuns:\n");
37             scanf("%d", &numGuns);
38
39             printf("Enter name of bore:\n");
40             scanf("%d", &bore);
41
42             printf("Enter name of displacement:\n");
43             scanf("%d", &displacement);
44
45             printf("Enter name of ship (if first ship, skip by
     pressing ENTER):\n");
46             fgets(shipName, sizeof shipName,stdin);
47
48             if (shipName[0] == '\n') {
49                 strncpy(shipName, class, sizeof(class));
50             }
51
52             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)"
```

14

```
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             SQLExecute(execStat);
65
66             SQLPrepare(execStat,
67                     "INSERT INTO Ships(name, class, launched)"
68                     "VALUES (?, ?, ?)", SQL_NTS);
69                 SQLBindParameter(execStat, 1, ..., shipName, ...);
70                 SQLBindParameter(execStat, 2, ..., class, ...);
71                 SQLBindParameter(execStat, 3, ..., dateLaunched, ...)
    ;
72             SQLExecute(execStat);
73         }
74     }
75
```

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, errorCode4;
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, &
     execStat);
33            errorCode4 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &
     subExecStat);
34        }
35
36        if (!errorCode3 && !errorCode4) {
37
38            SQLPrepare(execStat,
39                        "SELECT Ships.name, Ships.class, Outcomes.
     battle"
40                        "FROM Ships "
41                        "INNER JOIN Outcomes ON Ships.name = Outcomes
     .ship "
42                        "INNER JOIN Battles ON Outcomes.battle =
     Battles.name "
43                        "WHERE Ships.launched > Battles.date",
     SQL_NTS);
44            SQLExecute(execStat);
45            SQLBindCol(execStat, 1, SQL_CHAR, &shipName, sizeof(
     shipName), &shipNameInfo);
46            SQLBindCol(execStat, 2, SQL_FLOAT, &class, sizeof(class),
      &classInfo);
47            SQLBindCol(execStat, 3, SQL_INTEGER, &battle, sizeof(
     battle), &battleInfo);
48
49            while(SQLFetch(execStat) != SQL_NO_DATA) {
50
51                printf("Error. Ship %s is launched after date of
     battle.\n");
52
53                printf("Enter correct launched date (YYYY-MM-DD. Type
      0-0-0 to skip):\n");
54                scanf("%d-%d-%d", &newDateLaunchedDay, &
     newDateLaunchedMonth, &newDateLaunchedYear);
55
56                if (!newDateLaunchedDay == 0 &&
57                    !newDateLaunchedMonth == 0 &&
58                    !newDateLaunchedYear == 0) {
59                    // Correct date of launch
60                    SQLPrepare(subExecStat,
61                            "UPDATE Ships "
62                            "SET launched = ? "
63                            "WHERE name = ? AND class = ?",
     SQL_NTS);
64                        SQLBindParameter(subExecStat, 1, ...,
     newDateLaunched, ...);
65                        SQLBindParameter(subExecStat, 2, ...,
     shipName, ...);
66                        SQLBindParameter(subExecStat, 3, ..., class,
     ...);
67                    SQLExecute(subExecStat);
68                }
69
```

16

```
70              printf("Enter correct launched date (YYYY-MM-DD. Type
   0-0-0 to skip):\n");
71              scanf("%d-%d-%d", &newDateBattleDay, &
   newDateBattleMonth, &newDateBattleYear);
72
73          if (!newDateBattleDay == 0 &&
74              !newDateBattleMonth == 0 &&
75              !newDateBattleYear == 0) {
76              // Correct date of battle
77              SQLPrepare(subExecStat,
78                          "UPDATE Battles "
79                          "SET date = ? "
80                          "WHERE name = ?", SQL_NTS);
81                  SQLBindParameter(subExecStat, 1, ...,
   newDateBattle, ...);
82                  SQLBindParameter(subExecStat, 2, ..., battle,
    ...);
83              SQLExecute(subExecStat);
84          }
85      }
86    }
87  }
88
```

3. a)
```
 1   import java.sql.*;
 2   import java.util.Scanner;
 3
 4   class Q3a {
 5       public static void main(String[ ] args) {
 6
 7           int model, ram, hd, modelSol;
 8           float targetPrice, minDiff, speedSol, minDiff = Float.
   MAX_VALUE;
 9
10           String makerSol;
11
12           try {
13               // The newInstance() call is a work around for some
14               // broken Java implementations
15
16               Class.forName("com.mysql.jdbc.Driver").newInstance();
17               Connection myCon = DriverManager.getConnection("jdbc:
   mysql://localhost/Q3");
18               Statement execStat = myCon.createStatement();
19
20               Scanner scanObj = new Scanner(System.in);
21               System.out.println("Enter target price:");
22
23               String targetPriceRaw = scanObj.nextLine();  // Read
   user input
24               float targetPrice = Float.parseFloat(targetPriceRaw);
25
```

```
26              ResultSet results = execStat.executeQuery("SELECT
   model, speed, ram, hd, price, maker FROM Product NATURAL JOIN PC")
   ;
27
28          while (results.next()) {
29              model = results.getInt(1);
30              speed = results.getFloat(2);
31              ram = results.getFloat(3);
32              hd = results.getInt(4);
33              price = results.getInt(5);
34              maker = results.getString(6);
35
36              if (abs(price - targetPrice) >= minDiff) {
37                  continue;
38              }
39
40              minDiff = abs(price - targetPrice);
41              modelSol = model;
42              speedSol = speed;
43              makerSol = maker;
44          }
45
46          System.out.println(String.format("maker=%c, model=%d,
    speed=%.2f\n", makerSol, modelSol, speedSol));
47
48      }
49
50      } catch (SQLException ex) {
51          // handle the error
52          System.out.println("Error occured while establishing
   database connection");
53      }
54   }
55  }
56
```

Notes:

- JDBC
    - Setup

        1. Import JDBC

            import java.sql.*;

        2. Load a drive rfor the database system to use (i.e. sqlite, postgresql, mysql)

            Class.forName(< driver name >);

            Example:

```
1      Class.forName("com.mysql.jdbc.Driver");
2
```

3. Establish a connection to the database

   Connection myCon = DriverManager.getConnection(< URL >, < user name >, < password >);

   ∗ The url for jdbc is

   jdbc:mysql://< host name >/< database name >

– Creating Statements in JDBC
   1. **createStatement():**
      ∗ Prepares a statement object
      ∗ Is similar to **SQLAllocHandle** in C
   2. **prepareStatement(Q):**
      ∗ Prepares a statement object but with query
      ∗ Is similar to **SQLAllocHandle** and **SQLPrepare** combined in C
   3. **executeQuery(Q):**
      ∗ takes a query statement Q and executes it.
      ∗ Used for SELECT
      ∗ Is similar to **SQLPrepare** and **SQLExecute**
   4. **executeQuery():**
      ∗ Is used with **prepareStatement(Q)**
      ∗ Used for SELECT
      ∗ Is similar to **SQLExecute** in C
   5. **execUpdate(U):**
      ∗ takes a non-query statement U and executes it.
      ∗ Used for UPDATE AND INSERT
      ∗ Is similar to **SQLPrepare** and **SQLExecute**
   6. **executeQuery():**
      ∗ Is used with **prepareStatement(Q)**
      ∗ Used for UPDATE AND INSERT
      ∗ Is similar to **SQLExecute** in C

   Example:

```
1      // Example 1
2      Statement execStat = myCon.createStatement();
3      ResultSet worths = execStat.executeQuery(
4          "SELECT netWorth FROM MovieExec");
5
```

```
6     // Example 2
7     PreparedStatement execStat = myCon.prepareStatement(
8         "SELECT netWorth FROM MovieExec");
9     ResultSet worths = execStat.executeQuery()
10
11    // Example 3
12    Statement starStat = myCon.createStatement();
13    starStat.executeUpdate("INSERT INTO StarsIn VALUES('
      Remember the Titans', 2000, 'Denzel Washington')");
14
15    // Example 4
16    PreparedStatement starStat = myCon.prepareStatement(
17        "INSERT INTO StarsIn VALUES('Remember the Titans',
      2000, 'Denzel Washington')");
18    starStat.executeUpdate()");
19
```

– Cursor Operations in JDBC
  * **next():**
    · Moves to next tuple
  * **getString(i), getInt(i), getFloat(i):**
    · Fetches and converts value at ith column in tuple

  Example:

```
1     Statement execStat = myCon.createStatement();
2     ResultSet worths = execStat.executeQuery(
3         "SELECT netWorth FROM MovieExec");
4
5     while (worths.next()) {
6         int worth = worths.getInt(1);
7         ...
8     }
9
```

– Parameter Passing
  * **Syntax:** < PreparedStatement object >.setString(< column position >, < insert variable >);
  * Is similar to **SQLBindCol** in C

  Example:

```
1     PreparedStatement studioStat = myCon.prepareStatement(
2         "INSERT INTO Studio(name, address) VALUES(?, ?)");
3
4     studioStat.setString(1, studioName);
5     studioStat.setString(2, studioAddr);
6     studioStat.executeUpdate();
7
```

b)

```java
import java.util.Scanner;

class Q3b {
    public static void main(String[ ] args) {
        float minSpeed
        float minPrice
        float speed;
        float price;
        int minRam;
        int minHd;
        int model;
        int ram;
        int hd;
        int screen;
        String maker;

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            Connection myCon = DriverManager.getConnection("jdbc:
mysql://localhost/Q3");

            Scanner scanObj = new Scanner(System.in);

            System.out.println("Enter minimum speed:");
            String minSpeedRaw = scanObj.nextLine();
            minSpeed = Float.parseFloat(minSpeedRaw);

            System.out.println("Enter minimum ram:");
            String minRamRaw = scanObj.nextLine();
            minRam = Integer.parseFloat(minRamRaw);

            System.out.println("Enter minimum hard-drive space:")
;
            String hdRaw = scanObj.nextLine();
            hd = Integer.parseFloat(hdRaw);

            System.out.println("Enter minimum price:");
            String priceRaw = scanObj.nextLine();
            hd = Integer.parseFloat(priceRaw);

            PreparedStatement execStat = myCon.prepareStatement(
                "SELECT model, speed, ram, hd, screen, price,
maker" +
                "FROM Product NATURAL JOIN Laptop");
            ResultSet results = execStat.executeQuery();

            while(results.next()) {
                model = results.getInt(1);
                speed = results.getFloat(2);
                ram = results.getFloat(3);
                hd = results.getInt(4);
                screen = results.getInt(5);
                price = results.getInt(6);
                maker = results.getString(7);
```

```
52
53                        if (
54                            speed >= minSpeed &&
55                            ram >= minRam &&
56                            hd >= minHd &&
57                            screen >= minScreen
58                        ) {
59                            System.out.println(String.format("model=%d,
     speed=%.2f, ram=%d, hd=%d, screen=%d, price=%.2f, maker=%c",
60                                model, speed, ram, hd, screen, price,
     maker));
61                        }
62                    }
63
64            } catch (SQLException ex) {
65                    // handle the error
66                    System.out.println("Error occured while establishing
     database connection");
67            }
68        }
69    }
70
```

```
c)    import java.util.Scanner;
2
3     class Q3c {
4         public static void main(String[ ] args) {
5
6                int model;
7                boolean color;
8                String printType;
9                float price;
10
11                float speed;
12                int ram;
13                int hd;
14                int screen;
15                int productModel;
16                String productType;
17                String targetMaker;
18                String maker;
19
20
21            try {
22                    Class.forName("com.mysql.jdbc.Driver").newInstance();
23                    Connection myCon = DriverManager.getConnection("jdbc:
     mysql://localhost/Q3");
24                    Statement execStat = myCon.createStatement();
25
26                    System.out.println("Enter minimum speed:");
27                    targetMaker = scanObj.nextLine();
28
29                    ResultSet results = execStat.executeQuery("SELECT
     maker, productType FROM Product" +
```

```
30                                                          "GROUP BY
    maker, productType");
31
32                while (results.next()) {
33                    maker = results.getString(1);
34                    productType = results.getString(2);
35
36                    if (maker.toLowerCase() != targetMaker.
    toLowerCase()) continue;
37
38                    if (productType == 'pc') {
39                        PreparedStatement subExecStat = myCon.
    prepareStatement("SELECT speed, ram, hd, price FROM PC " +
40
                "NATURAL JOIN Product " +
41
                "WHERE type= ?");
42                            subExecStat.setString(1, productType);
43                        ResultSet subResults = subExecStat.
    executeQuery();
44
45                        while(subResults.next()) {
46                            speed = results.getFloat(1);
47                            ram = results.getFloat(2);
48                            hd = results.getInt(3);
49                            price = results.getInt(4);
50
51                            System.out.println(String.format("model=%
    d, speed=%.2f, ram=%d, hd=%d, price=%.2f, maker=%c, type=%s",
52                            model, speed, ram, hd, screen, price,
    maker, productType));
53                        }
54                    } else if (productType == 'laptop') {
55                        PreparedStatement subExecStat = myCon.
    prepareStatement("SELECT speed, ram, hd, screen, price FROM Laptop
     " +
56
                "NATURAL JOIN Product " +
57
                "WHERE type= ?");
58                            subExecStat.setString(1, productType);
59                        ResultSet subResults = subExecStat.
    executeQuery();
60
61                        while(subResults.next()) {
62                            speed = results.getFloat(1);
63                            ram = results.getFloat(2);
64                            hd = results.getInt(3);
65                            screen = results.getInt(4);
66                            price = results.getInt(5);
67
68                            System.out.println(String.format("model=%
    d, speed=%.2f, ram=%d, hd=%d, screen=%d, price=%.2f, maker=%c,
    type=%s",
```

23

```
69                              model, speed, ram, hd, screen, screen,
     price, maker, productType));
70                              }
71
72                      } else if (strcmp(productType, 'printer')) {
73                          PreparedStatement subExecStat = myCon.
     prepareStatement("SELECT color, printType, price FROM Printer " +
74
                 "NATURAL JOIN Product " +
75
                 "WHERE type= ?");
76                              subExecStat.setString(1, productType);
77                          ResultSet subResults = subExecStat.
     executeQuery();
78
79                          while(subResults.next()) {;
80                              color = results.getBoolean(1);
81                              printType = results.getString(2);
82                              price = results.getFloat(3);
83
84                              printf("model=%d, color=%s, price=%.2f,
     maker=%c, type=%s",
85                              model, color ? "true" : "false", price,
     maker, type);
86                          }
87                      }
88                  }
89
90          } catch (SQLException ex) {
91              // handle the error
92              System.out.println("Error occured while establishing
     database connection");
93          }
94      }
95   }
96
```

d)

e)
```
     import java.util.Scanner;
2
     class Q3c {
3
4        public static void main(String[ ] args) {
5
6            int model;
7            boolean color;
8            String printType;
9            float price;
10
11           float speed;
12           int ram;
13           int hd;
14           int screen;
15           int productModel;
16           String productType;
```

```
17              String targetMaker;
18              String maker;
19

20

21          try {
22                  Class.forName("com.mysql.jdbc.Driver").newInstance();
23                  Connection myCon = DriverManager.getConnection("jdbc:
    mysql://localhost/Q3");
24                  Statement execStat = myCon.createStatement();
25

26                  System.out.println("Enter minimum speed:");
27                  targetMaker = scanObj.nextLine();
28

29                  ResultSet results = execStat.executeQuery("SELECT
    maker, productType FROM Product" +
30                                                      "GROUP BY
     maker, productType");
31

32                  while (results.next()) {
33                      maker = results.getString(1);
34                      productType = results.getString(2);
35

36                      if (maker.toLowerCase() != targetMaker.
    toLowerCase()) continue;
37

38                      if (productType == 'pc') {
39                          PreparedStatement subExecStat = myCon.
    prepareStatement("SELECT speed, ram, hd, price FROM PC " +
40              "NATURAL JOIN Product " +
41              "WHERE type= ?");
42                              subExecStat.setString(1, productType);
43                          ResultSet subResults = subExecStat.
    executeQuery();
44

45                          while(subResults.next()) {
46                              speed = results.getFloat(1);
47                              ram = results.getFloat(2);
48                              hd = results.getInt(3);
49                              price = results.getInt(4);
50

51                              System.out.println(String.format("model=%
    d, speed=%.2f, ram=%d, hd=%d, price=%.2f, maker=%c, type=%s",
52                              model, speed, ram, hd, screen, price,
    maker, productType));
53                          }
54                      } else if (productType == 'laptop') {
55                          PreparedStatement subExecStat = myCon.
    prepareStatement("SELECT speed, ram, hd, screen, price FROM Laptop
     " +
56              "NATURAL JOIN Product " +
57
```

```
                  "WHERE type= ?");
58                            subExecStat.setString(1, productType);
59                        ResultSet subResults = subExecStat.
     executeQuery();
60
61                        while(subResults.next()) {
62                            speed = results.getFloat(1);
63                            ram = results.getFloat(2);
64                            hd = results.getInt(3);
65                            screen = results.getInt(4);
66                            price = results.getInt(5);
67
68                            System.out.println(String.format("model=%
     d, speed=%.2f, ram=%d, hd=%d, screen=%d, price=%.2f, maker=%c,
     type=%s",
69                            model, speed, ram, hd, screen, screen,
     price, maker, productType));
70                        }
71
72                    } else if (strcmp(productType, 'printer')) {
73                        PreparedStatement subExecStat = myCon.
     prepareStatement("SELECT color, printType, price FROM Printer " +
74
               "NATURAL JOIN Product " +
75
               "WHERE type= ?");
76                            subExecStat.setString(1, productType);
77                        ResultSet subResults = subExecStat.
     executeQuery();
78
79                        while(subResults.next()) {;
80                            color = results.getBoolean(1);
81                            printType = results.getString(2);
82                            price = results.getFloat(3);
83
84                            printf("model=%d, color=%s, price=%.2f,
     maker=%c, type=%s",
85                            model, color ? "true" : "false", price,
     maker, type);
86                        }
87                    }
88                }
89
90            } catch (SQLException ex) {
91                // handle the error
92                System.out.println("Error occured while establishing
     database connection");
93            }
94        }
95    }
96
```

f)

g)   `import java.util.Scanner;`

```
2
3      class Q3e {
4          public static void main(String[ ] args) {
5
6              int model, ram, hd;
7              float speed, price;
8              String maker;
9
10             try {
11                 Class.forName("com.mysql.jdbc.Driver").newInstance();
12                 Connection myCon = DriverManager.getConnection("jdbc:
    mysql://localhost/Q3");
13
14                 Scanner scanObj = new Scanner(System.in);
15
16                 System.out.println("Enter manufacturer:");
17                 maker = scanObj.nextLine();
18
19                 System.out.println("Enter model:");
20                 String modelRaw = scanObj.nextLine();
21                 model = Integer.parseInt(modelRaw);
22
23                 System.out.println("Enter speed:");
24                 String speedRaw = scanObj.nextLine();
25                 speed = Float.parseFloat(speedRaw);
26
27                 System.out.println("Enter ram:");
28                 String ramRaw = scanObj.nextLine();
29                 ram = Float.parseFloat(ramRaw);
30
31                 System.out.println("Enter hd:");
32                 String hdRaw = scanObj.nextLine();
33                 hd = Float.parseFloat(hdRaw);
34
35                 System.out.println("Enter price:");
36                 String priceRaw = scanObj.nextLine();
37                 price = Float.parseFloat(priceRaw);
38
39
40                 PrepareStatement execStat = myCon.prepareStatement("
    SELECT Product.model AS m1, PC.model AS m2 FROM " +
41                                                               "
    Product FULL OUTER JOIN PC" +
42                                                               "
    WHERE model = ?");
43                     execStat.setString(1, model);
44                 ResultSet results = execStat.executeQuery();
45
46                 while (results.next()) {
47                     productModel = results.getString(1);
48                     pcModel = results.getString(2);
49
50                     if (productModel != null & pcModel != null) {
51                         System.out.println("Error. Model already
```

```
       exists in database.");
  52                   } else {
  53                       PreparedStatement subExecStat = myCon.
       prepareStatement(
  54                           "INSERT INTO PC(model, speed, ram, hd,
       price) " +
  55                           "VALUES(?, ?, ?, ?, ?)"
  56                       );
  57                       subExecStat.setString(1, model);
  58                       subExecStat.setString(2, speed);
  59                       subExecStat.setString(3, ram);
  60                       subExecStat.setString(4, hd);
  61                       subExecStat.setString(5, price);
  62                   subExecStat.executeUpdate();
  63
  64                   subExecStat = myCon.prepareStatement(
  65                       "INSERT INTO Product(model, maker, type)
       " +
  66                       "VALUES(?, ?, 'pc')"
  67                   );
  68                       subExecStat.setString(1, model);
  69                       subExecStat.setString(2, maker);
  70                   subExecStat.executeUpdate();
  71               }
  72
  73           }
  74       } catch (SQLException ex) {
  75           // handle the error
  76           System.out.println("Error occured while establishing
       database connection");
  77       }
  78     }
  79   }
  80
```

4. a)
```
   import java.sql.*;
  2  import java.util.Scanner;
  3
  4  class Q4a {
  5      public static void main(String[ ] args) {
  6          int class;
  7
  8          try {
  9              Class.forName("com.mysql.jdbc.Driver").newInstance();
 10              Connection myCon = DriverManager.getConnection("jdbc:
       mysql://localhost/Q3");
 11
 12              Scanner scanObj = new Scanner(System.in);
 13
 14              PrepareStatement execStat = myCon.prepareStatement("
       SELECT class FROM FROM Classes " +
 15                                                                 "
       WHERE numGuns * POWER(bore, 3) >= ALL ( " +
```

```
16                                                                              "
     SELECT numGuns * POWER(bore, 3) FROM Classes)");
17
18
     ResultSet results = execStat.executeQuery();
19
20              while (results.next()) {
21                  class = results.getString(1);
22                  System.out.println(String.format("Class = %s\n",
     class));
23              }
24
25
26          } catch (SQLException ex) {
27              // handle the error
28              System.out.println("Error occured while establishing
     database connection");
29          }
30      }
31   }
32
```

Correct Solution:

```
1              import java.sql.*;
2              import java.util.Scanner;
3
4              class Q4a {
5                  public static void main(String[ ] args) {
6                      int class;
7
8                      try {
9                          Class.forName("com.mysql.jdbc.Driver").
   newInstance();
10                         Connection myCon = DriverManager.
   getConnection("jdbc:mysql://localhost/Q4"); // <- Corrected
11
12                         Scanner scanObj = new Scanner(System.in);
13
14                         PrepareStatement execStat = myCon.
   prepareStatement("SELECT class FROM FROM Classes " +
15              "WHERE numGuns * POWER(bore, 3) >= ALL ( " +
16              "SELECT numGuns * POWER(bore, 3) FROM Classes)");
17
18              ResultSet results = execStat.executeQuery();
19
20                      while (results.next()) {
21                          class = results.getString(1);
```

```
22                                    System.out.println(String.format("
     Class = %s\n", class));
23                                    }
24
25
26                      } catch (SQLException ex) {
27                            // handle the error
28                            System.out.println("Error occured while
     establishing database connection");
29                      }
30                  }
31              }
32
```

b)
```
1     import java.sql.*;
2     import java.util.Scanner;
3
4
5     class Q4b {
6         public static void main(String[ ] args) {
7             String targetBattle;
8             String country;
9             int count;
10
11            String mostSunkCountry;
12            int maxSunkCount = 0;
13
14            String mostDamagedCountry;
15            int maxDamagedCount = 0;
16
17            try {
18                Class.forName("com.mysql.jdbc.Driver").newInstance();
19                Connection myCon = DriverManager.getConnection("jdbc:
     mysql://localhost/Q4"); // <- Corrected
20
21                Scanner scanObj = new Scanner(System.in);
22
23                Scanner scanObj = new Scanner(System.in);
24                System.out.println("Enter name of battle:");
25                targetBattle = scanObj.nextLine();
26
27                PrepareStatement sunkStat = myCon.prepareStatement("
     SELECT country, COUNT(Outcomes.result) FROM Classes " +
28                                                                   "
     INNER JOIN Ships ON Classes.class = Ships.class " +
29                                                                   "
     INNER JOIN Outcomes ON Ships.name = Outcomes.ship " +
30                                                                   "
     INNER JOIN Battles ON Battles.name = Outcome.battle " +
31                                                                   "
     GROUP BY country " +
32                                                                   "
     HAVING Battles.name= ? AND " +
33                                                                   "
```

```
      Outcomes.result='sunk'");
34                    sunkStat.setString(1, targetBattle);
35                ResultSet sunkResults = sunkStat.executeQuery();
36
37                while(sunkResults.next()) {
38                    country = sunkResults.getString(1);
39                    count = sunkResults.getInt(2);
40
41                    if (count > maxSunkCount) {
42                        maxSunkCount = count;
43                        mostSunkCountry = country;
44                    }
45                }
46
47                printf("Country with most sunk ships: %s",
      mostSunkCountry);
48
49                PrepareStatement damagedStat = myCon.prepareStatement
      ("SELECT country, COUNT(Outcomes.result) FROM Classes " +
50                                                                     "
      INNER JOIN Ships ON Classes.class = Ships.class " +
51                                                                     "
      INNER JOIN Outcomes ON Ships.name = Outcomes.ship " +
52                                                                     "
      INNER JOIN Battles ON Battles.name = Outcome.battle " +
53                                                                     "
      GROUP BY country " +
54                                                                     "
      HAVING Battles.name= ? AND " +
55                                                                     "
      Outcomes.result='damaged'");
56
57                    sunkStat.setString(1, targetBattle);
58                ResultSet damagedResults = damagedStat.executeQuery()
      ;
59
60                while(damagedResults.next()) {
61                    country = damagedResults.getString(1);
62                    count = damagedResults.getInt(1);
63
64                    if (count > maxDamagedCount) {
65                        maxDamagedCount = count;
66                        mostDamagedCountry = country;
67                    }
68                }
69
70                printf("Country with most damaged ships: %s",
      mostDamagedCountry);
71
72
73          } catch (SQLException ex) {
74              // handle the error
75              System.out.println("Error occured while establishing
      database connection");
```

```
76                }
77
78            }
79        }
80


c)      import java.sql.*;
2       import java.util.Scanner;
3
4
5       class Q4c {
6           public static void main(String[ ] args) {
7               try {
8                   Class.forName("com.mysql.jdbc.Driver").newInstance();
9                   Connection myCon = DriverManager.getConnection("jdbc:
    mysql://localhost/Q4");
10
11                  Scanner scanObj = new Scanner(System.in);
12
13                  System.out.println("Enter name of class:");
14                  String class = scanObj.nextLine();
15
16                  System.out.println("Enter name of type ('bb' or 'bc')
    :");
17                  String type = scanObj.nextLine();
18
19                  System.out.println("Enter name of country:");
20                  String country = scanObj.nextLine();
21
22                  System.out.println("Enter name of numGuns:");
23                  String numGunsRaw = scanObj.nextLine();
24                  numGuns = Integer.parseFloat(numGunsRaw);
25
26                  System.out.println("Enter name of bore:");
27                  String boreRaw = scanObj.nextLine();
28                  bore = Integer.parseFloat(boreRaw);
29
30                  System.out.println("Enter name of displacement:");
31                  String displacementRaw = scanObj.nextLine();
32                  displacement = Integer.parseFloat(displacementRaw);
33
34                  System.out.println("Enter name of ship (if first ship
    , skip by pressing ENTER):");
35                  String shipName = scanObj.nextLine();
36
37                  if (shipName.trim().equals("")) {
38                      shipName = class;
39                  }
40
41                  System.out.println("Enter date launched (YYYY-MM-DD):
    ");
42                  String dateLaunched = scanObj.nextLine();
43
44
```

```
45                      PreparedStatement classExecStat = myCon.
      prepareStatement(
46                          "INSERT INTO Classes(class, type, country,
      numGuns, bore, displacement) " +
47                          "VALUES (?, ?, ?, ?, ?, ?)"
48                      );
49                          classExecStat.setString(1, class);
50                          classExecStat.setString(2, type);
51                          classExecStat.setString(3, country);
52                          classExecStat.setString(4, numGuns);
53                          classExecStat.setString(5, bore);
54                          classExecStat.setString(6, displacement);
55                      classExecStat.executeUpdate();
56
57                      PreparedStatement shipsExecStat = myCon.
      prepareStatement(
58                          "INSERT INTO Ships(name, class, launched) " +
59                          "VALUES (?, ?, ?)"
60                      );
61                          shipsExecStat.setString(1, shipName);
62                          shipsExecStat.setString(2, class);
63                          shipsExecStat.setString(3, dateLaunched);
64                      shipsExecStat.executeUpdate();
65
66              } catch (SQLException ex) {
67                  // handle the error
68                  System.out.println("Error occured while establishing
      database connection");
69              }
70          }
71      }
72
```

5. a)
```
    <?php
2   include(DB.php);
3
4   function Q5a() {
5       $targetPrice = INF;
6
7       $modelSol;
8       $speedSol;
9       $makerSol;
10
11      $myCon = DB::connect(mysql://root:dbPassword@localhost/Q5);
12
13      $stdin = fopen('php://stdin', 'r');
14      echo("Enter target price:");
15      $targetPrice = fscanf(STDIN, "%f\n");
16
17      $results = $myCon->query("SELECT model, speed, ram, hd, price
    , maker FROM Product NATURAL JOIN PC");
18
19      while ($tuple = $results->fetchRow()) {
20          $model = $tuple[0];
```

```php
21          $speed = $tuple[1];
22          $ram = $tuple[2];
23          $hd = $tuple[3];
24          $price = $tuple[4];
25          $maker = $tuple[5];
26
27          if (abs($price - $targetPrice) >= $minDiff) {
28              continue;
29          }
30
31          $minDiff = abs($price - $targetPrice);
32          $modelSol = $model;
33          $speedSol = $speed;
34          $makerSol = $maker;
35      }
36
37      sprintf("maker=%c, model=%d, speed=%.2f\n", makerSol,
    modelSol, speedSol);
38      $myCon->disconnect();
39  }
40  ?>
41
```

Notes:

- PHP basics
  - code is put inside a special tag

    <?php
    PHP code goes in here
    ? >
- Variables
  - starts with '$'.
- String
  - is surrounded by single quote or double quote
  - single quote → literal string
  - double quote → can replace special symbols inside quote with values

    ### Example:

    ```php
    1    // Here, $ in single quote is printed as is
    2    $foo = 'bar';
    3    $x = 'step up to the $foo'
    4
    5    // Here, $ in double quote is replaced with corresponding
         value of variable
    6    $foo = "bar";
    7    $x = "step up to the $foo" // <- prints "step up to the bar
         "
    8
    ```

  – concatenates two strings using dot

```
1       $foo = "foo" . " bar"; // <- $foo has value "foo bar";
2
```

- Arrays
  - **Syntax 1:** array(< val 1 >, < val 2 >, < val 3 >, ...)
  - **Syntax 2:** array(< key 1 > => < val 1 >, < key 2 > => < val 2 >, < key 3 > => < val 3 >, ...)

```
1       $seasons = array('spring' => 'warm')
2
```

- Connecting to Database
  - **Syntax:** $myCon = DB::connect(< vendor >://< user name >:< password > @ < host name >/< database name >);
  - connection to server disconnect via DB_Object_Instance− > disconnect()
- Executing SQL statements
  - **Syntax:** DB_Object_Instance− > query(statement)

  Example:

```
1       $result = $myCon->query("INSERT INTO StarsIn VALUES("
2           'Denzel Washington', 2000, 'Remember the Titans'")")
3
```

- Cursor Operations in PHP

  Example:

```
1      $worths = $myCon->query("SELECT netWorth FROM MovieExec");
2      while ($tuple = $worhts->fetchRow()) {
3          $worth = $tuple[0];
4      }
5
```

- Dynamic SQL in PHP

  Example:

```
1      $worths = $myCon->prepare("INSERT INTO Studio(name, address)
       VALUES(?,?)");
2      $args = array('MGM', 'Los Angles');
3      $result = $myCon->execute($prepQuery, $args);
4
```

b)
```php
<?php
function Q5b() {

    $myCon = DB::connect(mysql://root:dbPassword@localhost/Q5);

    echo("Enter minimum speed:");
    $minSpeed = fscanf(STDIN, "%f\n");

    echo("Enter minimum ram:");
    $minRam = fscanf(STDIN, "%d\n");

    echo("Enter minimum hard-drive space:");
    $minHd = fscanf(STDIN, "%d\n");

    echo("Enter minimum price:");
    $minPrice = fscanf(STDIN, "%f\n");

    $results = $myCon->query("SELECT model, speed, ram, hd,
screen, price, maker" +
                            "FROM Product NATURAL JOIN Laptop");

    while($tuple = $results->fetchRow()) {
        $model = $tuple[0];
        $speed = $tuple[1];
        $ram = $tuple[2];
        $hd = $tuple[3];
        $screen = $tuple[4];
        $price = $tuple[5];
        $maker = $tuple[6];

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

    $myCon->disconnect();
}
```

c)
```php
<?php
function Q5c() {

    $myCon = DB::connect(mysql://root:dbPassword@localhost/Q5);

    echo("Enter maker:");
    $targetMaker = fscanf(STDIN, "%s\n");

```

```
9          $results = $myCon->query("SELECT maker, productType FROM
   Product" +
10                               "GROUP BY maker, productType");
11
12
13         while ($tuple = $results->fetchRow()) {
14             $maker = $tuple[0];
15             $productType = $tuple[1];
16
17             if (strtolower(maker) != strtolowertargetMaker)))
   continue;
18
19             if (productType == 'pc') {
20                 $subQuery = $myCon->prepare("SELECT speed, ram, hd,
   price FROM PC " +
21                                             "NATURAL JOIN Product " +
22                                             "WHERE type= ?");
23
24                 $args = array($productType);
25                 $subResults = $myCon->execute($subQuery, $args);
26
27                 while($tuple = $subResults->fetchRow()) {
28                     $speed = $tuple[0];
29                     $ram = $tuple[1];
30                     $hd = $tuple[2];
31                     $price = $tuple[3];
32
33                     sprintf("model=%d, speed=%.2f, ram=%d, hd=%d,
   price=%.2f, maker=%c, type=%s",
34                     model, speed, ram, hd, screen, price, maker,
   productType);
35                 }
36             } else if (productType == 'laptop') {
37                 $subQuery = $myCon->prepare("SELECT speed, ram, hd,
   screen, price FROM Laptop " +
38                                             "NATURAL JOIN Product " +
39                                             "WHERE type= ?");
40                 $args = array($productType);
41                 $subResults = $myCon->execute($subQuery, $args);
42
43                 while($tuple = $subResults->fetchRow()) {
44                     $speed = $tuple[0];
45                     $ram = $tuple[1];
46                     $hd = $tuple[2];
47                     $screen = $tuple[3];
48                     $price = $tuple[4];
49
50                     sprintf("model=%d, speed=%.2f, ram=%d, hd=%d,
   screen=%d, price=%.2f, maker=%c, type=%s",
51                     model, speed, ram, hd, screen, screen, price,
   maker, productType);
52                 }
53
54             } else if (strcmp(productType, 'printer')) {
```

37

```
55
56                 $subQuery = $myCon->prepare("SELECT color, printType,
     price FROM Printer " +
57                                      "NATURAL JOIN Product " +
58                                      "WHERE type= ?");
59             $args = array($productType);
60             $subResults = $myCon->execute($subQuery, $args);
61
62             while($tuple = $subResults->fetchRow()) {;
63                 $color = $tuple[1];
64                 $printType = $tuple[2];
65                 $price = $tuple[3];
66
67                 sprintf("model=%d, color=%s, price=%.2f, maker=%c
     , type=%s",
68                 model, color ? "true" : "false", price, maker,
     type);
69             }
70         }
71     }
72
73     $myCon->disconnect();
74  }
75
```

d)

e)
```
    function Q5e {
2
3       $myCon = DB::connect(mysql://root:dbPassword@localhost/Q5);
4
5       echo("Enter manufacturer:");
6       $maker = fscanf(STDIN, "%f\n");
7
8       System.out.println("Enter model:");
9       String modelRaw = scanObj.nextLine();
10      model = Integer.parseInt(modelRaw);
11
12      echo("Enter model:");
13      $model = fscanf(STDIN, "%d\n");
14
15      echo("Enter speed:");
16      $speed = fscanf(STDIN, "%f\n");
17
18      echo("Enter ram:");
19      $ram = fscanf(STDIN, "%d\n");
20
21      echo("Enter hd:");
22      $hd = fscanf(STDIN, "%d\n");
23
24      echo("Enter price:");
25      $price = fscanf(STDIN, "%f\n");
26
27      $execStat = $myCon->prepare("SELECT Product.model AS m1, PC.
    model AS m2 FROM " +
```

```
28                                            "Product FULL OUTER JOIN PC" +
29                                            "WHERE model = ?");
30                 $args = array(model);
31             $results = $myCon->execute();
32
33             while ($tuple = $results->fetchRow()) {
34                 $productModel = $tuple[0];
35                 $pcModel = $tuple[1];
36
37                 if (!is_null($productModel) && !is_null($pcModel)) {
38                     echo("Error. Model already exists in database.");
39                 } else {
40                     $subExecStat = $myCon->prepare(
41                         "INSERT INTO PC(model, speed, ram, hd, price) " +
42                         "VALUES(?, ?, ?, ?, ?)"
43                     );
44                         $args = array(model, speed, ram, hd, price);
45                     $myCon->execute($subExecStat, $args);
46
47                     $subExecStat = $myCon->statement(
48                         "INSERT INTO Product(model, maker, type) " +
49                         "VALUES(?, ?, 'pc')"
50                     );
51                         $args = array(model, maker);
52                     $myCon->execute($subExecStat, $args);
53                 }
54
55             }
56         }
57
58
```

6. a)
```
1     <?php
2     function Q6a () {
3         $myCon = DB::connect(mysql://root:dbPassword@localhost/Q6);
4
5         $results = $myCon->query("SELECT class FROM FROM Classes " +
6                                  "WHERE numGuns * POWER(bore, 3) >=
    ALL ( " +
7                                  "SELECT numGuns * POWER(bore, 3)
    FROM Classes)");
8
9         while ($tuple = $results->fetchRow()) {
10             class = $tuple[1];
11             sprintf("Class = %s\n", class);
12         }
13     }
14     ?>
15
```

b)
```
1     <?php
2     function Q6b () {
3         $maxSunkCount;
```

```
 4          $mostSunkCountry;
 5          $maxDamagedCount;
 6          $mostDamagedCountry;
 7          $myCon = DB::connect(mysql://root:dbPassword@localhost/Q6);
 8
 9          Scanner scanObj = new Scanner(System.in);
10          System.out.println("Enter name of battle:");
11          targetBattle = scanObj.nextLine();
12
13          $query = $myCon->prepare("SELECT country, COUNT(Outcomes.
    result) FROM Classes " +
14                                   "INNER JOIN Ships ON Classes.class =
     Ships.class " +
15                                   "INNER JOIN Outcomes ON Ships.name =
     Outcomes.ship " +
16                                   "INNER JOIN Battles ON Battles.name
    = Outcome.battle " +
17                                   "GROUP BY country " +
18                                   "HAVING Battles.name= ? AND " +
19                                   "Outcomes.result='sunk'");
20
21          $args = array(targetBattle);
22          $sunkResults = $myCon->execute($query, $args);
23
24
25          while ($tuple = $results->fetchRow()) {
26              $country = $tuple[0];
27              $count = $tuple[1];
28
29              if ($count > $maxSunkCount) {
30                  $maxSunkCount = $count;
31                  $mostSunkCountry = $country;
32              }
33          }
34
35          sprintf("Country with most sunk ships: %s", mostSunkCountry);
36
37          $query = $myCon->prepare("SELECT country, COUNT(Outcomes.
    result) FROM Classes " +
38                                   "INNER JOIN Ships ON Classes.class =
     Ships.class " +
39                                   "INNER JOIN Outcomes ON Ships.name =
     Outcomes.ship " +
40                                   "INNER JOIN Battles ON Battles.name
    = Outcome.battle " +
41                                   "GROUP BY country " +
42                                   "HAVING Battles.name= ? AND " +
43                                   "Outcomes.result='damaged'");
44          $args = array(targetBattle);
45          $damagedResults = $myCon->execute($query, $args);
46
47
48          while($tuple = $results->fetchRow()) {
49              $country = $tuple[0];
```

```
50            $count = $tuple[1];
51
52            if ($count > $maxDamagedCount) {
53                $maxDamagedCount = $count;
54                $mostDamagedCountry = $country;
55            }
56        }
57
58        sprintf("Country with most damaged ships: %s",
    mostDamagedCountry);
59     }
60     ?>
61
```

c)
```
   <?php
 2     function Q6c () {
 3         $myCon = DB::connect(mysql://root:dbPassword@localhost/Q5);
 4
 5         echo("Enter name of class:");
 6         $class = $_POST['class'];
 7
 8         echo("Enter name of type ('bb' or 'bc'):");
 9         $type = $_POST['type'];
10
11         echo("Enter name of country:");
12         $country = $_POST['country'];
13
14         echo("Enter name of numGuns:");
15         $numGuns = $_POST['numGuns'];
16
17         echo("Enter name of bore:");
18         $bore = $_POST['bore'];
19
20         echo("Enter name of displacement:");
21         $displacement = $_POST['displacement'];
22
23         echo("Enter name of ship (if first ship, skip by pressing
    ENTER):");
24         $shipName = $_POST['shipName'];
25
26         if (trim($shipName) == "") {
27             shipName = $_POST['class'];
28         }
29
30         $dateLaunched = $_POST['dateLaunched'];
31
32
33         PreparedStatement classExecStat = myCon.prepareStatement(
34             "INSERT INTO Classes(class, type, country, numGuns, bore,
    displacement) " +
35             "VALUES (?, ?, ?, ?, ?, ?)"
36         );
37             classExecStat.setString(1, class);
38             classExecStat.setString(2, type);
```

```
39            classExecStat.setString(3, country);
40            classExecStat.setString(4, numGuns);
41            classExecStat.setString(5, bore);
42            classExecStat.setString(6, displacement);
43        classExecStat.executeUpdate();
44
45        PreparedStatement shipsExecStat = myCon.prepareStatement(
46            "INSERT INTO Ships(name, class, launched) " +
47            "VALUES (?, ?, ?)"
48        );
49            shipsExecStat.setString(1, shipName);
50            shipsExecStat.setString(2, class);
51            shipsExecStat.setString(3, dateLaunched);
52        shipsExecStat.executeUpdate();
53
54
55        $myCon->disconnect();
56
57    }
58    ?>
59
```

d)
```
1    <?php
2    function Q6d () {
3
4    }
5    ?>
6
```