

9.3.1

a)

In the following, we use macro NOT_FOUND as defined in the section.

```
void closestMatchPC() {

    EXEC SQL BEGIN DECLARE SECTION;

        char manf, SQLSTATE[6];
        int targetPrice, /* holds price given by user */
        float tempSpeed, speedOfClosest;
        char tempModel[4], modelOfClosest[4];
        int tempPrice, priceOfClosest;
        /* for tuple just read from PC & closest price found so far */

    EXEC SQL END DECLARE SECTION;

    EXEC SQL DECLARE pcCursor CURSOR FOR
        SELECT model, price, speed FROM PC;

    EXEC SQL OPEN pcCursor;

    /* ask user for target price and read the answer into variable
       targetPrice */

    /* Initially, the first PC is the closest to the target price.
       If PC is empty, we cannot answer the question, and so abort. */

    EXEC SQL FETCH FROM pcCursor INTO :modelOfClosest, :priceOfClosest,
                                         :speedOfClosest;
    if(NOT_FOUND) /* print message and exit */ ;

    while(1) {
        EXEC SQL FETCH pcCursor INTO :tempModel, :tempPrice,
                                         :tempSpeed;

        if(NOT_FOUND) break;
        if(/*tempPrice closer to targetPrice than is priceOfClosest */)
        {
            modelOfClosest = tempModel;
            priceOfClosest = tempPrice;
            speedOfClosest = tempSpeed;
        }
    }

    /* Now, modelOfClosest is the model whose price is closest to
       target. We must get its manufacturer with a single-row select */

    EXEC SQL SELECT maker
        INTO :manf
        FROM Product
        WHERE model = :modelOfClosest;

    printf("manf=%s, model=%d, speed=%d\n",
           manf, modelOfClosest, speedOfClosest);

    EXEC SQL CLOSE CURSOR pcCursor;
```

```
}
```

b)

```
void acceptableLaptop() {
```

```
    EXEC SQL BEGIN DECLARE SECTION;
```

```
        int minRam, minHd, minScreen; /* given by user */
        float minSpeed;
        char model[4], maker,
        float speed;
        int ram, hd, screen, price;
```

```
    EXEC SQL END DECLARE SECTION;
```

```
    EXEC SQL PREPARE query1 FROM
```

```
        'SELECT model, speed, ram, hd, screen, price, maker
          FROM Laptop l, Product p
          WHERE speed >= ? AND
                ram >= ? AND
                hd >= ? AND
                screen >= ? AND
                l.model = p.model'
```

```
    EXEC SQL DECLARE cursor1 CURSOR FOR query1;
```

```
    /* ask user for minimum speed, ram, hd size, and screen size */
```

```
    EXEC SQL OPEN cursor1 USING :minSpeed, :minRam, :minHd, :minScreen;
```

```
    while(!NOT_FOUND) {
```

```
        EXEC SQL FETCH cursor1 INTO
            :model, :speed, :ram, :hd, :screen, :price, :maker;
```

```
        if(FOUND)
```

```
        {
            printf("maker:%s, model:%d, \n
                  speed:%.2f, ram:%d, hd:%d, screen:%d, price:%d\n",
                  maker, model, speed, ram, hd, screen, price);
        }
```

```
    }
```

```
    EXEC SQL CLOSE CURSOR cursor1;
```

```
}
```

c)

```
void productsByMaker() {
```

```
    EXEC SQL BEGIN DECLARE SECTION;
```

```
        char maker, model[4], type[10], color[6];
        float speed;
        int ram, hd, screen, price;
```

```

EXEC SQL END DECLARE SECTION;

EXEC SQL PREPARE query1 FROM
    'SELECT * FROM PC
      WHERE model IN (SELECT model FROM Product
                      WHERE maker = ? AND
                      type = 'pc');

EXEC SQL PREPARE query2 FROM
    'SELECT * FROM Laptop
      WHERE model IN (SELECT model FROM Product
                      WHERE maker = ? AND
                      type = 'laptop');

EXEC SQL PREPARE query3 FROM
    'SELECT * FROM Printer
      WHERE model IN (SELECT model FROM Product
                      WHERE maker = ? AND
                      type = 'printer');

EXEC SQL DECLARE cursor1 CURSOR FOR query1;
EXEC SQL DECLARE cursor2 CURSOR FOR query2;
EXEC SQL DECLARE cursor3 CURSOR FOR query3;

/* ask user for manufacturer */
Printf("maker:%s\n", maker);

/* get PCs made by the manufacturer */
EXEC SQL OPEN cursor1 USING :maker;

Printf("product type: PC\n");

while(!NOT_FOUND) {

    EXEC SQL FETCH cursor1 INTO
        :model, :speed, :ram, :hd, :price;

    if(FOUND)
    {
        printf("model:%d,speed:%.2f, ram:%d, hd:%d, price:%d\n",
            model, speed, ram, hd, price);
    }
}

/* get Laptops made by the manufacturer */
EXEC SQL OPEN cursor2 USING :maker;

Printf("product type: Laptop\n");

while(!NOT_FOUND) {

    EXEC SQL FETCH cursor2 INTO
        :model, :speed, :ram, :hd, :screen, :price;

    if(FOUND)
    {
        printf("model:%d, speed:%.2f, ram:%d, hd:%d, screen:%d,
            price:%d\n", model, speed, ram, hd, screen, price);
    }
}

```

```

    }
}

/* get Printers made by the manufacturer */
EXEC SQL OPEN cursor3 USING :maker;

Printf("product type: Printer\n");

while(!NOT_FOUND) {

    EXEC SQL FETCH cursor3 INTO
        :model, :color, :type, :price;

    if(FOUND)
    {
        printf("model:%d, color:%s, type:%s, price:%d\n",
            model, color, type, price);
    }
}

EXEC SQL CLOSE CURSOR cursor1;
EXEC SQL CLOSE CURSOR cursor2;
EXEC SQL CLOSE CURSOR cursor3;
}

d)

void withinBudget() {

    EXEC SQL BEGIN DECLARE SECTION;

    int total_budget, rest_budget, pc_price, printer_price;
    char pc_model[4], printer_model[4], color[6];
    float min_speed;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL PREPARE query1 FROM
        'SELECT model, price FROM PC
        WHERE speed >= ? AND price <= ?
        ORDER BY price';

    EXEC SQL PREPARE query2 FROM
        'SELECT model, price FROM Printer
        WHERE price <= ? AND color = ?
        ORDER BY price';

    EXEC SQL DECLARE cursor1 CURSOR FOR query1;
    EXEC SQL DECLARE cursor2 CURSOR FOR query2;

    /* ask user for budget & the minimum speed of pc */

    /* get the cheapest PC of the minimum speed */
    EXEC SQL OPEN cursor1 USING :min_speed, :total_budget;

```

```

EXEC SQL FETCH cursor1 INTO :pc_model, :pc_price;

if (NOT_FOUND)
    Printf("no pc found within the budget\n");
else
{
    Printf("pc model: %s\n", pc_model);
}

/* get Printer within the budget */
rest_budget = total_budget - pc_price;
color = "true";

EXEC SQL OPEN cursor2 USING :rest_budget, :color;

EXEC SQL FETCH cursor2 INTO :printer_model;

if(NOT_FOUND) {
    EXEC SQL CLOSE CURSOR cursor2;
    color = "false";
    EXEC SQL OPEN cursor2 USING :rest_budget, :color;

    if(NOT_FOUND)
        printf("no printer found within the budget\n");
    else {
        EXEC SQL FETCH cursor2 INTO :printer_model;
        printf("printer model: %s\n", printer_model);
    }
}
else {
    printf("printer model: %s\n", printer_model);
}

EXEC SQL CLOSE CURSOR cursor1;
EXEC SQL CLOSE CURSOR cursor2;
}

```

e)

```

void newPCproduct() {

    EXEC SQL BEGIN DECLARE SECTION;

        char pmaker, pmodel[4], ptype[6];
        float pspeed;
        int pram, phd, pscreen, pprice;
        int pcount;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL PREPARE stmt1 FROM
        'SELECT COUNT(*) INTO :count
          FROM PC
          WHERE MODEL = ?;

    EXEC SQL PREPARE stmt2 FROM
        'INSERT INTO Product VALUES(?, ?, ?)';
}

```

```

EXEC SQL PREPARE stmt3 FROM
    'INSERT INTO PC VALUES(?, ?, ?, ?, ?)';

/* ask user for manufacturer, model, speed, RAM, hard-disk,
   & price of a new PC */

EXEC SQL EXECUTE stmt1 USING :pmodel;

IF (count > 0)
    Printf("Warnning: The PC model already exists\n");
ELSE
{
    EXEC SQL EXECUTE stmt2 USING :pmaker, :pmodel, :ptype;

    EXEC SQL EXECUTE stmt3 USING :pmodel, :pspeed, :pram,
                                  :phd, :pprice
}
}

```

9.3.2

a)

```

void largestFirepower() {

    EXEC SQL BEGIN DECLARE SECTION;

        char cclass[20], maxFirepowerClass[20];
        int cnumGuns, cbore;
        float firepower, maxFirepower;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL DECLARE cursor1 CURSOR FOR
        SELECT class, numGuns, bore FROM Classes;

    EXEC SQL OPEN cursor1;

    EXEC SQL FETCH FROM cursor1 INTO :cclass, :cnumGuns, :cbore;

    if(NOT_FOUND) /* print message and exit */ ;

    maxFirepower = cnumGuns * (power (cbore, 3));
    strcpy(maxFirepowerClass, cclass);

    while(1) {
        EXEC SQL FETCH cursor1 INTO :cclass, :cnumGuns, :cbore;

        if(NOT_FOUND) break;

        firepower = cnumGuns * (power (cbore, 3));

        if( firepower > maxFirepower )
        {
            maxFirepower = firepower;

```

```

        strcpy(maxFirepowerClass, cclass);
    }
}

printf("Class of maximum firepower :%s\n", maxFirepowerClass);

EXEC SQL CLOSE CURSOR cursor1;
}

b)

void getCountry() {

    EXEC SQL BEGIN DECLARE SECTION;

        char ibattle[20], iresult[10], ocountry[20];
        char stmt1[200], stmt2[200];

    EXEC SQL END DECLARE SECTION;

    strcpy(stmt1, "SELECT COUNTRY FROM Classes C
        WHERE C.class IN (
            SELECT S.class FROM Ships S
            WHERE S.name IN (
                SELECT ship FROM Outcomes
                WHERE battle = ?))");

    Strcpy(stmt2, "SELECT country FROM Classes
        WHERE class = ( SELECT MAX(COUNT(class))
            FROM Ships s, Outcomes o
            WHERE o.name = s.ship AND
            s.result = '?' )");

    EXEC SQL PREPARE query1 FROM stmt1;
    EXEC SQL PREPARE query2 FROM stmt2;

    EXEC SQL DECLARE cursor1 CURSOR FOR query1;
    EXEC SQL DECLARE cursor2 CURSOR FOR query2;

    /* ask user for battle */

    /* get countries of the ships involved in the battle */
    EXEC SQL OPEN cursor1 USING :ibattle;

    while(!NOT_FOUND) {

        EXEC SQL FETCH cursor1 INTO :ocountry;

        if(FOUND)
            printf("contry:%s\n", ocountry);
    }

    EXEC SQL CLOSE CURSOR cursor1;

    /* get the country with the most ships sunk */
    strcpy(iresult, "sunk");

```

```

EXEC SQL OPEN cursor2 USING :iresult;

/* loop for the case there's the same max# of ships sunk */
While(!NOT_FOUND) {

    EXEC SQL FETCH cursor2 INTO :ocountry;

    If(FOUND)
        Printf("country with the most ships sunk: %s, ocountry);

}

/* get the country with the most ships damaged */
strcpy(iresult, "damaged");

EXEC SQL OPEN cursor2 USING :iresult;

/* loop for the case there's the same max# of ships damaged */
While(!NOT_FOUND) {

    EXEC SQL FETCH cursor2 INTO :ocountry;

    If(FOUND)
        Printf("country with the most ships damaged: %s, ocountry);

}

}

c)

void addShips() {

    EXEC SQL BEGIN DECLARE SECTION;

        char iclass[20], itype[3], icontry[20], iship[20];
        int inumGuns, ibore, idisplacement, ilaunched;
        char stmt1[100], stmt2[100];

    EXEC SQL END DECLARE SECTION;

    strcpy(stmt1, "INSERT INTO Classes VALUES (?, ?, ?, ?, ?, ?)");
    strcpy(stmt2, "INSERT INTO Ships VALUES (?, ?, ?)");

    /* ask user for a class and other info for Classes table */

    EXEC SQL EXECUTE IMMEDIATE :stmt1
        USING :iclass, :itype, :icontry,
            :inumGuns, :ibore, :idisplacement;

    /* ask user for a ship and launched */

    WHILE(there_is_input)
    {

```



```

        EXEC SQL EXECUTE IMMEDIATE :stmt2
            USING :iship, :iclass, ilaunched;

        /* ask user for a ship and launched */
    }

}

d)

void findError() {

    EXEC SQL BEGIN DECLARE SECTION;

        char bname[20], bdate[8], newbdate[8];
        char sname[20], lyear[4], newlyear[4];
        char stmt1[100], stmt2[100];

    EXEC SQL END DECLARE SECTION;

    strcpy(stmt1, "UPDATE Battles SET date = ? WHERE name = ?");
    strcpy(stmt2, "UPDATE Ships SET launched = ? WHERE name = ?");

    EXEC SQL DECLARE C1 CURSOR FOR
        Select b.name, b.date, s.name, s.launched
        FROM Battles b, Outcomes o, Ships s
        WHERE b.name = o.battle AND
              o.ship = s.name AND
              YEAR(b.date) < s.launched;

    EXEC SQL OPEN C1;

    while(!NOT_FOUND) {

        EXEC SQL FETCH C1 INTO :bname, :bdate, :sname, :lyear;

        /* prompt user and ask if a change is needed */

        if(change_battle)
        {
            /* get a new battle date to newbdate */
            EXEC SQL EXECUTE IMMEDIATE :stmt1
                USING :bname, :newbdate;
        }

        if(change_ship)
        {
            /* get a new launched year to newlyear */
            EXEC SQL EXECUTE IMMEDIATE :stmt2
                USING :sname, :newlyear;
        }
    }
}

```

9.4.1

a)

```
CREATE FUNCTION PresNetWorth(studioName CHAR[15]) RETURNS INTEGER
```

```
DECLARE presNetWorth INT;
```

```
BEGIN
```

```
    SELECT netWorth
    INTO presNetWorth
    FROM Studio, MovieExec
    WHERE Studio.name = studioName AND presC# = cert#;
    RETURN(presNetWorth);
```

```
END;
```

b)

```
CREATE FUNCTION status(person CHAR(30), addr CHAR(255)) RETURNS INTEGER
```

```
DECLARE isStar INT;
```

```
DECLARE isExec INT;
```

```
BEGIN
```

```
    SELECT COUNT(*)
    INTO isStar
    FROM MovieStar
    WHERE MovieStar.name = person AND MovieStar.address = addr;
    SELECT COUNT(*)
    INTO isExec
    FROM MovieExec
    WHERE MovieExec.name = person AND MovieExec.address = addr;
    IF isStar + isExec = 0 THEN RETURN(4)
    ELSE RETURN(isStar + 2*isExec)
    END IF;
```

```
END;
```

c)

```
CREATE PROCEDURE twoLongest(
```

```
    IN studio CHAR(15),
    OUT longest VARCHAR(255),
    OUT second VARCHAR(255)
```

```
)
```

```
DECLARE t VARCHAR(255);
```

```
DECLARE i INT;
```

```
DECLARE Not_Found CONDITION FOR SQLSTATE = '02000';
```

```
DECLARE MovieCursor CURSOR FOR
```

```
    SELECT title FROM Movies WHERE studioName = studio
    ORDER BY length DESC;
```

```
BEGIN
```

```
    SET longest = NULL;
```

```
    SET second = NULL;
```

```
    OPEN MovieCursor;
```

```
    SET i = 0;
```

```

        mainLoop: WHILE (i < 2) DO
            FETCH MovieCursor INTO t;
            IF Not_Found THEN LEAVE mainLoop END IF;
            SET i = i + 1;
        END WHILE;
        CLOSE MovieCursor;
    END;

```

d)

```

CREATE PROCEDURE earliest120mMovie(
    IN star CHAR(30),
    OUT earliestYear INT
)

DECLARE Not_Found CONDITION FOR SQLSTATE = '02000';
DECLARE MovieCursor CURSOR FOR
    SELECT MIN(year) FROM Movies
        WHERE length > 120 AND
            title IN (SELECT movieTitle FROM StarsIn
                WHERE starName = star);

BEGIN
    SET earliestYear = 0;
    OPEN MovieCursor;
    FETCH MovieCursor INTO earliestYear;
    CLOSE MovieCursor;
END;

```

e)

```

CREATE PROCEDURE uniqueStar(
    IN addr CHAR(255),
    OUT star CHAR(30)
)

BEGIN
    SET star = NULL;
    IF 1 = (SELECT COUNT(*) FROM MovieStar WHERE address = addr)
    THEN
        SELECT name INTO star FROM MovieStar WHERE address = addr;
    END;

```

f)

```

CREATE PROCEDURE removeStar(
    IN star CHAR(30)
)

BEGIN
    DELETE FROM Movies WHERE title IN
        (SELECT movieTitle FROM StarsIn WHERE starName = star);
    DELETE FROM StarsIn WHERE starName = star;
    DELETE FROM MovieStar WHERE name = star;
END;

```

9.4.2

a)

```
CREATE FUNCTION closestMatchPC(targetPrice INT) RETURNS CHAR

DECLARE closestModel CHAR(4);
DECLARE diffSq INT;
DECLARE currSq INT;
DECLARE m CHAR(4);
DECLARE p INT;
DECLARE Not_Found CONDITION FOR SQLSTATE '02000';
DECLARE PCCursor CURSOR FOR
    SELECT model, price FROM PC;

BEGIN
    SET closestModel = NULL;
    SET diffSq = -1;
    OPEN PCCursor;
    mainLoop: LOOP
        FETCH PCCursor INTO m, p;
        IF Not_Found THEN LEAVE mainLoop END IF;
        SET currSq = (p - targetPrice)*(p - targetPrice);
        IF diffSq = -1 OR diffSq > currSq
            THEN BEGIN
                SET closestModel = m;
                SET diffSq = currSq;
            END IF;
        END LOOP;
    CLOSE PCCursor;
    RETURN(closestModel);
END;
```

b)

```
CREATE FUNCTION getPrice(imaker CHAR(1), imodel CHAR(4))
    RETURNS INTEGER

DECLARE ptype VARCHAR(10);
DECLARE pprice INT;
DECLARE Not_Found CONDITION FOR SQLSTATE '02000';

BEGIN
    SELECT type INTO ptype FROM Product
        WHERE maker = imaker AND model = imodel;
    IF ptype = 'pc' THEN
        SELECT price INTO pprice FROM PC
            WHERE model = imodel;
    ELSE IF ptype = 'laptop' THEN
        SELECT price INTO pprice FROM Laptop
            WHERE model = imodel;
    ELSE IF ptype = 'printer' THEN
        SELECT price INTO pprice FROM Printer
            WHERE model = imodel;
    ELSE
        pprice = NULL;
    END IF;
```

```
        RETURN (pprice);
END;
```

c)

```
CREATE PROCEDURE addPC(
    IN imodel INT,
    IN ispeed DECIMAL(3,2),
    IN iram INT,
    IN ihd INT,
    IN iprice INT
)

DECLARE Already_Exist CONDITION FOR SQLSTATE '02300';

BEGIN

    INSERT INTO PC VALUES(imodel, ispeed, iram, ihd, iprice);
    WHILE (Already_Exist) DO
        SET imodel = imodel + 1;
        INSERT INTO PC VALUES(imodel, ispeed, iram, ihd, iprice);
    END WHILE;
END;
```

d)

```
CREATE PROCEDURE getNumOfHigherPrice(
    IN iprice INT,
    OUT NumOfPCs INT,
    OUT NumOfLaptops INT,
    OUT NumOfPrinters INT
)

BEGIN
    SET NumOfPCs = 0;
    SET NumOfLaptops = 0;
    SET NumOfPrinters = 0;

    SELECT COUNT(*) INTO NumOfPCs FROM PC
        WHERE price > iprice;

    SELECT COUNT(*) INTO NumOfLaptops FROM Laptop
        WHERE price > iprice;

    SELECT COUNT(*) INTO NumOfPrinters FROM Printer
        WHERE price > iprice;
END;
```

9.4.3

a)

```
CREATE FUNCTION getFirepower(iclass VARCHAR(10)) RETURNS INTEGER
```

```
DECLARE firepower INT;
```

```
DECLARE nguns INT;
```

```
DECLARE nbore INT;
```

```
BEGIN
```

```
    SELECT numGuns, bore INTO nguns, nbore FROM Classes  
        WHERE class = iclass;
```

```
    SET firepower = nguns * (nbore * nbore * nbore);
```

```
    RETURN(firepower);
```

```
END;
```

b)

```
CREATE PROCEDURE twoCountriesInBattle(
```

```
    IN ibattle VARCHAR(20),
```

```
    OUT firstCountry VARCHAR(20),
```

```
    OUT secondCountry VARCHAR(20)
```

```
)
```

```
DECLARE i INT;
```

```
DECLARE ocountry VARCHAR(20);
```

```
DECLARE classCursor CURSOR FOR
```

```
    SELECT country FROM Classes
```

```
        WHERE class IN(SELECT class FROM Ships
```

```
            WHERE name IN(
```

```
                SELECT ship FROM Outcomes
```

```
                WHERE battle = ibattle
```

```
            )
```

```
        );
```

```
BEGIN
```

```
    SET firstCountry = NULL;
```

```
    SET secondCountry = NULL;
```

```
    SET i = 0;
```

```
    IF 2 = (SELECT COUNT(*) count FROM Classes
```

```
        WHERE class IN(SELECT class FROM Ships
```

```
            WHERE name IN(
```

```
                SELECT ship FROM Outcomes
```

```
                WHERE battle = ibattle
```

```
            )
```

```
        )
```

```
    )
```

```
THEN
```

```
    OPEN classCursor;
```

```
    WHILE (i < 2) DO
```

```
        FETCH classCursor INTO ocountry;
```

```
        IF (i = 0) THEN
```

```
            SET firstCountry = ocountry;
```

```
        ELSE
```

```
            SET secondCountry = ocountry;
```

```
        END IF;
```

```

        END WHILE;
    END IF;
    CLOSE calssCursor;
END;

```

c)

```

CREATE PROCEDURE addClass(
    IN iship VARCHAR(20),
    IN iclass VARCHAR(20),
    IN itype CHAR(2),
    IN icountry VARCHAR(20),
    IN inumGuns INT,
    IN ibore INT,
    IN idisplacement INT
)
BEGIN
    INSERT INTO Classes VALUES(iclass, itype, icountry,
                                inumGuns, ibore, idisplacement);
    INSERT INTO Ships VALUES(iship, iclass, NULL);
END;

```

d)

```

CREATE PROCEDURE checkLaunched(
    IN ship VARCHAR(20)
)
DECLARE bname VARCHAR(20);
BEGIN
    IF EXIST (SELECT b.name INTO bname
              FROM Battles b, Outcomes o, Ships s
              WHERE b.name = o.battle AND
                    o.ship = s.name AND
                    YEAR(b.date) < s.launched)
    THEN
        UPDATE Ships SET launced = 0 WHERE name = iship;
        UPDATE Battles SET date = 0 WHERE name = bname;
    END IF;
END

```

9.4.4

$$\begin{aligned}
\left[\sum_{i=1}^n (x_i - \bar{x})^2 \right] / n &= 1/n \left[\sum_{i=1}^n (x_i^2 - 2\bar{x}x_i + \bar{x}^2) \right] \\
&= 1/n \left[\sum x_i^2 - \sum 2\bar{x}x_i + \sum \bar{x}^2 \right] \\
&= 1/n \left[\sum x_i^2 - 2\bar{x} \sum x_i + \bar{x}^2 \sum 1 \right] \\
&= 1/n \left[\sum x_i^2 - 2\bar{x}(n\bar{x}) + \bar{x}^2(n) \right] \\
&= 1/n \left[\sum x_i^2 - n\bar{x}^2 \right] \quad , \text{ since } \bar{x} = \sum_{i=1}^n (x_i) / n \\
&= \left[\sum_{i=1}^n (x_i)^2 \right] / n - \left[\left(\sum_{i=1}^n x_i \right) / n \right]^2
\end{aligned}$$

9.5.1

a)

```
#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR manf, tempModel[4];
SQLFLOAT tempSpeed;
SQLINTEGER tempPrice;
SQLINTEGER colInfo;

Int targetPrice;
char modelOfClosest[4];
float speedOfClosest;
int priceOfClosest;

/* ask user for target price and read the answer into variable
   targetPrice */

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

SQLExecDirect(execStat, "SELECT model, price, speed FROM PC", SQL_NTS);

SQLBindCol(execStat, 1, SQL_CHAR, tempModel,
            sizeof(tempModel), &colInfo);
SQLBindCol(execStat, 2, SQL_INTEGER, tempPrice,
            sizeof(tempPrice), &colInfo);
SQLBindCol(execStat, 3, SQL_FLOAT, tempSpeed,
            sizeof(tempSpeed), &colInfo);

priceOfClosest = NULL;
while(SQLFetch(execStat) != SQL_NO_DATA) {

    if( /* the 1st fetch or tempPrice closer to targetPrice */
        modelOfClosest = tempModel;
```

```

        priceOfClosest = tempPrice;
        speedOfClosest = tempSpeed;
    }

}

/* Now, modelOfClosest is the model whose price is closest to
   target. We must get its manufacturer with a single-row select
*/

if (priceOfClosest == NULL ) /* no data fetched */
    /* print error message and exit */

SQLPrepare (execStat,
            "SELECT maker FROM Product WHERE model = ?", SQL_NTS);
SQLBindParameter(execStat, 1,...,modelOfClosest, ...);
SQLExecute(execStat);
SQLBindCol(execStat, 1, SQLCHAR, &manf, sizeof(manf), &colInfo);

/* print manf */

b)

#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR model[4], maker;
SQLFLOAT minSpeed;
SQLINTEGER minRam, minHd, minScreen;
SQLFLOAT speed;
SQLINTEGER ram, hd, screen;
SQLINTEGER colInfo;

/* ask user for minimum speed, ram, hd size, and screen size */

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

```

```

SQLPrepare(execStat,
    "SELECT model, speed, ram, hd, screen, price, maker " ||
    "FROM Laptop l, Product p " ||
    "WHERE speed >= ? AND " ||
    "ram >= ? AND " ||
    "hd >= ? AND " ||
    "screen >= ? AND " ||
    "l.model = p.model",
    SQL_NTS);

SQLBindParameter(execStat, 1, SQL_FLOAT, ..., minSpeed, ...);
SQLBindParameter(execStat, 2, SQL_INTEGER, ..., minRam, ...);
SQLBindParameter(execStat, 3, SQL_INTEGER, ..., minHd, ...);
SQLBindParameter(execStat, 4, SQL_INTEGER, ..., minScreen, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_CHAR, model, sizeof(model), &colInfo);
SQLBindCol(execStat, 2, SQL_FLOAT, speed,
    sizeof(speed), &colInfo);
SQLBindCol(execStat, 3, SQL_INTEGER, ram,
    sizeof(ram), &colInfo);
SQLBindCol(execStat, 4, SQL_INTEGER, hd,
    sizeof(hd), &colInfo);
SQLBindCol(execStat, 5, SQL_INTEGER, screen,
    sizeof(screen), &colInfo);
SQLBindCol(execStat, 6, SQL_INTEGER, price,
    sizeof(price), &colInfo);
SQLBindCol(execStat, 7, SQL_CHAR, maker,
    sizeof(maker), &colInfo);

while(SQLFetch(execStat) != SQL_NO_DATA) {
    if( FOUND )
        /* print fetched info */
}

c)

#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR maker, model[4], type[10], color[6];
SQLFLOAT speed;
SQLINTEGER ram, hd, screen, price;
SQLINTEGER colInfo;

/* ask user for minimum speed, ram, hd size, and screen size */

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

```

```

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_ HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_ HANDLE_STMT.\n");
    exit(1);
}

/* get PCs made by the manufacturer */

SQLPrepare(execStat,
            "SELECT * FROM PC WHERE model IN (" ||
            "SELECT model FROM Product " ||
            "WHERE maker = ? AND " ||
            "type = 'pc'",
            SQL_NTS);

SQLBindParameter(execStat, 1, SQL_CHAR, ..., maker, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_CHAR, model, sizeof(model), &colInfo);
SQLBindCol(execStat, 2, SQL_FLOAT, speed,
            sizeof(speed), &colInfo);
SQLBindCol(execStat, 3, SQL_INTEGER, ram,
            sizeof(ram), &colInfo);
SQLBindCol(execStat, 4, SQL_INTEGER, hd,
            sizeof(hd), &colInfo);
SQLBindCol(execStat, 5, SQL_INTEGER, price,
            sizeof(price), &colInfo);

while(SQLFetch(execStat) != SQL_NO_DATA) {

    if( FOUND )
        /* print fetched info */
    }

/* get Laptops made by the manufacturer */

SQLPrepare(execStat,
            "SELECT * FROM Laptop WHERE model IN (" ||
            "SELECT model FROM Product " ||
            "WHERE maker = ? AND " ||
            "type = 'laptop'",
            SQL_NTS);

SQLBindParameter(execStat, 1, SQL_CHAR, ..., maker, ...);

SQLExecute(execStat);

```

```

SQLBindCol(execStat, 1, SQL_CHAR, model, sizeof(model), &colInfo);
SQLBindCol(execStat, 2, SQL_FLOAT, speed,
            sizeof(speed), &colInfo);
SQLBindCol(execStat, 3, SQL_INTEGER, ram,
            sizeof(ram), &colInfo);
SQLBindCol(execStat, 4, SQL_INTEGER, hd,
            sizeof(hd), &colInfo);
SQLBindCol(execStat, 5, SQL_INTEGER, screen,
            sizeof(screen), &colInfo);
SQLBindCol(execStat, 6, SQL_INTEGER, price,
            sizeof(price), &colInfo);

while(SQLFetch(execStat) != SQL_NO_DATA) {

    if( FOUND )
        /* print fetched info */
}

/* get Printers made by the manufacturer */

SQLPrepare(execStat,
            "SELECT * FROM Printer WHERE model IN (" ||
            "SELECT model FROM Product " ||
            "WHERE maker = ? AND " ||
            "type = 'printer'",

            SQL_NTS);

SQLBindParameter(execStat, 1, SQL_CHAR, ..., maker, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_CHAR, model, sizeof(model), &colInfo);
SQLBindCol(execStat, 2, SQL_CHAR, color, sizeof(color), &colInfo);
SQLBindCol(execStat, 3, SQL_CHAR, type, sizeof(type), &colInfo);
SQLBindCol(execStat, 4, SQL_INTEGER, price, sizeof(price), &colInfo);

d)

#include sqlcli.h
SQLHENV myEnv;
SQLHDBC myCon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLINTEGER total_budget, rest_budget, pc_price, printer_price;
SQLCHAR pc_model[4], printer_model[4], color[6];
SQLFLOAT min_speed;

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

```

```

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

SQLPrepare(execStat,
            "SELECT model, price FROM PC
             WHERE speed >= ? AND price <= ?
             ORDER BY price",
            SQL_NTS);

/* ask user for budget & the minimum speed of pc */

/* get the cheapest PC of the minimum speed */

SQLBindParameter(execStat, 1, SQL_FLOAT, ..., min_speed, ...);
SQLBindParameter(execStat, 2, SQL_INTEGER, ..., total_budget, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_CHAR, pc_model, sizeof(pc_model), &colInfo);
SQLBindCol(execStat, 2, SQL_INTEGER, pc_price,
            sizeof(pc_price), &colInfo);

SQLFetch(execStat);

if (NOT_FOUND) {
    printf("no pc found within the budget\n");
}
else {
    printf("pc model: %s\n", pc_model);
}

/* get Printer within the budget */
rest_budget = total_budget - pc_price;
color = "true";

SQLPrepare(execStat, "SELECT model, price FROM Printer
                     WHERE price <= ? AND color = ?
                     ORDER BY price", SQL_NTS);

SQLBindParameter(execStat, 1, SQL_INTEGER, ..., rest_budget, ...);
SQLBindParameter(execStat, 2, SQL_CHAR, ..., color, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_CHAR, print_model,
            sizeof(print_model), &colInfo);
SQLBindCol(execStat, 2, SQL_INTEGER, print_price,
            sizeof(print_price), &colInfo);

```

```

SQLFetch(execStat);

if(NOT_FOUND) {
    color = "false";

    SQLBindParameter(execStat, 1, SQL_INTEGER, ..., rest_budget, ...);
    SQLBindParameter(execStat, 2, SQL_CHAR, ..., color, ...);

    SQLExecute(execStat);

    SQLBindCol(execStat, 1, SQL_CHAR, print_model,
                sizeof(print_model), &colInfo);
    SQLBindCol(execStat, 2, SQL_INTEGER, print_price,
                sizeof(print_price), &colInfo);

    SQLFetch(execStat);

    if(NOT_FOUND)
        printf("no printer found within the budget\n");
    else
        printf("printer model: %s\n", printer_model);
}
else
    printf("printer model: %s\n", printer_model);
}

e)

#include sqlcli.h
SQLHENV myEnv;
SQLHDBC myCon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR pmodel[4], pmaker, ptype[6];
SQLFLOAT pspeed;
SQLINTEGER pram, phd, pscreen, pprice, count;
SQLINTEGER colInfo;

/* ask user for minimum speed, ram, hd size, and screen size */

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

```

```

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

/* ask user for manufacturer, model, speed, RAM, hard-disk, */
/* & price of a new PC */

SQLPrepare(execStat,
            "SELECT COUNT(*) FROM PC WHERE model = ?",
            SQL_NTS);

SQLBindParameter(execStat, 1, SQL_CHAR, ..., pmodel, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_INTEGER, count, sizeof(count), &colInfo);

SQLFetch(execStat);

if (count > 0) {
    Printf("Warning: The PC model already exists\n");
}
else {

    SQLPrepare(execStat, "INSERT INTO Product VALUES(?, ?, ?)",
                SQL_NTS);

    SQLBindParameter(execStat, 1, SQL_CHAR, ..., pmaker, ...);
    SQLBindParameter(execStat, 2, SQL_CHAR, ..., pmodel, ...);
    SQLBindParameter(execStat, 3, SQL_CHAR, ..., ptype, ...);
    SQLExecute(execStat);

    SQLPrepare(execStat, "INSERT INTO PC VALUES(?, ?, ?, ?, ?)",
                SQL_NTS);

    SQLBindParameter(execStat, 1, SQL_CHAR, ..., pmodel, ...);
    SQLBindParameter(execStat, 2, SQL_FLOAT, ..., pspeed, ...);
    SQLBindParameter(execStat, 3, SQL_INTEGER, ..., pram, ...);
    SQLBindParameter(execStat, 4, SQL_INTEGER, ..., phd, ...);
    SQLBindParameter(execStat, 5, SQL_INTEGER, ..., pprice, ...);
    SQLExecute(execStat);

}

```

9.5.2

a)

```

#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR cclass[20], maxFirepowerClass[20];
SQLFLOAT firepower, maxFirepower;
SQLINTEGER cnumGuns, cbore;

```



```

SQLINTEGER colInfo;

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

SQLExecDirect(execStat,
               "SELECT class, numGuns, bore FROM Classes", SQL_NTS);

SQLBindCol(execStat, 1, SQL_CHAR, cclass, sizeof(cclass), &colInfo);
SQLBindCol(execStat, 2, SQL_INTEGER, cnumGuns, sizeof(cnumGuns),
            &colInfo);
SQLBindCol(execStat, 3, SQL_INTEGER, cbore, sizeof(cbore), &colInfo);

SQLFetch(execStat);

if(NOT_FOUND) /* print message and exit */ ;

maxFirepower = cnumGuns * (power (cbore, 3));
strcpy(maxFirepowerClass, cclass);

while(1) {

    SQLFetch(execStat);

    if(NOT_FOUND) break;

    firepower = cnumGuns * (power (cbore, 3));

    if( firepower > maxFirepower )
    {
        maxFirepower = firepower;
        strcpy(maxFirepowerClass, cclass);
    }
}

printf("Class of maximum firepower :%s\n", maxFirepowerClass);

```

b)

```
#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR ibattle[20], ocountry[20];
SQLINTEGER colInfo;

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

SQLPrepare(execStat,
    "SELECT COUNTRY FROM Classes C WHERE C.class IN (
        SELECT S.class FROM Ships S WHERE S.name IN (
            SELECT ship FROM Outcomes WHERE battle = ?))",
    SQL_NTS);

SQLBindParameter(execStat, 1, SQL_CHAR, ..., ibattle, ...);

SQLExecute(execStat);

SQLBindCol(execStat, 1, SQL_CHAR, ocountry, sizeof(ocountry),
    &colInfo);

while(SQLFetch(execStat) != SQL_NO_DATA) {
    printf("contry:%s\n", ocountry);
}
```

c)

```
#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
```

```

SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR iclass[20], itype[3], icountry[20], iship[20];
SQLINTEGER inumGuns, ibore, idisplacement, ilaunched;
SQLINTEGER colInfo;

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

/* ask user for a class and other info for Classes table */

SQLPrepare(execStat,
           "INSERT INTO Classes VALUES (?, ?, ?, ?, ?, ?)", SQL_NTS);

SQLBindParameter(execStat, 1, SQL_CHAR, ..., iclass, ...);
SQLBindParameter(execStat, 2, SQL_CHAR, ..., itype, ...);
SQLBindParameter(execStat, 3, SQL_CHAR, ..., icountry, ...);
SQLBindParameter(execStat, 4, SQL_INTEGER, ..., inumGuns, ...);
SQLBindParameter(execStat, 5, SQL_INTEGER, ..., ibore, ...);
SQLBindParameter(execStat, 6, SQL_INTEGER, ..., idisplacement,...);

SQLExecute(execStat);

/* ask user for a ship and launched */

SQLPrepare(execStat, "INSERT INTO Ships VALUES (?, ?, ?)", SQL_NTS);

WHILE(there_is_input)
{
    SQLBindParameter(execStat, 1, SQL_CHAR, ..., iship, ...);
    SQLBindParameter(execStat, 2, SQL_CHAR, ..., iclass, ...);
    SQLBindParameter(execStat, 3, SQL_INTEGER, ..., ilaunched, ...);

    SQLExecute(execStat);

    /* ask user for a ship and launched */
}

d)

```

```

#include sqlcli.h
SQLHENV myEnv;
SQLHDBC mycon;
SQLHSTMT execStat;
SQLRETURN errCode1, errCode2, errCode3;
SQLCHAR bname[20], bdate[8], newbdate[8];
SQLCHAR sname[20], lyear[4], newlyear[4];
SQLINTEGER colInfo;

errCode1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &myEnv);
if(errCode1) {
    printf("Error for SQL_HANDLE_ENV.\n");
    exit(1);
}

errCode2 = SQLAllocHandle(SQL_HANDLE_DBC, myEnv, &myCon);

if(errCode2) {
    printf("Error for SQL_HANDLE_DBC.\n");
    exit(1);
}

errCode3 = SQLAllocHandle(SQL_HANDLE_STMT, myCon, &execStat);

if(errCode3) {
    printf("Error for SQL_HANDLE_STMT.\n");
    exit(1);
}

SQLExecDirect(execStat,
    "Select b.name, b.date, s.name, s.launched " ||
    "FROM Battles b, Outcomes o, Ships s " ||
    "WHERE b.name = o.battle AND " ||
    "o.ship = s.name AND " ||
    "YEAR(b.date) < s.launched ",
    SQL_NTS);

SQLBindCol(execStat, 1, SQL_CHAR, bname, sizeof(bname), &colInfo);
SQLBindCol(execStat, 2, SQL_CHAR, bdate, sizeof(bdate), &colInfo);
SQLBindCol(execStat, 3, SQL_CHAR, sname, sizeof(sname), &colInfo);
SQLBindCol(execStat, 4, SQL_CHAR, lyear, sizeof(lyear), &colInfo);

while(SQLFetch(execStat) != SQL_NO_DATA) {

    /* prompt user and ask if a change is needed */

    if(change_battle)
    {
        /* get a new battle date to newbdate */
        SQLPrepare(execStat, "UPDATE Battles SET date = ? WHERE name = ?",
            SQL_NTS);
        SQLBindParameter(execStat, 1, ..., newbdate, ...);
        SQLBindParameter(execStat, 2, ..., bname, ...);

        SQLExecute(execStat);
    }
}

```

```
}

if(change_ship)
{
    /* get a new launched year to newlyear */
    SQLPrepare(execStat, "UPDATE Ships SET launched = ? WHERE name= ?",
                SQL_NTS);
    SQLBindParameter(execStat, 1, ..., newlyear, ...);
    SQLBindParameter(execStat, 2, ..., sname, ...);

    SQLExecute(execStat);
}
}
```

9.6.1

a)

```
import java.sql.*;

char manf, tempModel[4];
float tempSpeed;
int tempPrice;

Int targetPrice;
char modelOfClosest[4];
float speedOfClosest;
int priceOfClosest;

Class.forName("<drive name>");

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

/* ask user for target price and read the answer into variable
   targetPrice */

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT model, price, speed FROM PC");
ResultSet pcs = execStat.executeQuery();

While(pcs.next()) {

    tempModel = pcs.getString(1);
    tempPrice = pcs.getInt(2);
    tempSpeed = pcs.getFloat(3);

    if( /* the 1st fetch or tempPrice closer to targetPrice */ ) {
        modelOfClosest = tempModel;
        priceOfClosest = tempPrice;
        speedOfClosest = tempSpeed;
    }

}

/* Now, modelOfClosest is the model whose price is closest to
   target. We must get its manufacturer with a single-row select
   */

if (priceOfClosest == NULL ) /* no data fetched */
    /* print error message and exit */

PreparedStatement execStat2 = myCon.prepareStatement(
    "SELECT maker FROM Product WHERE model = ?");
execStat2.setString(1, modelOfClosest);

ResultSet makers = execStat2.executeQuery();

/* print manf */
```

b)

```
import java.sql.*;

char model[4], maker;
float minSpeed;
int minRam, minHd, minScreen;
float speed;
int ram, hd, screen;

Class.forName("<drive name>");

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

/* ask user for minimum speed, ram, hd size, and screen size */

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT model, speed, ram, hd, screen, price, maker " +
    "FROM Laptop l, Product p " +
    "WHERE speed >= ? AND " +
    "       ram >= ? AND " +
    "       hd >= ? AND " +
    "       screen >= ? AND " +
    "       l.model = p.model");

execStat.setFloat(1, minSpeed);
execStat.setInt(2, minRam);
execStat.setInt(3, minHd);
execStat.setInt(4, minScreen);

ResultSet products = execStat.executeQuery();

While(products.next()) {

    model = getString(1);
    speed = getFloat(2);
    ram = getInt(3);
    hd = getInt(4);
    screen = getInt(5);
    price = getInt(6);
    maker = getString(7);

    /* print fetched info */

}
```

c)

```
import java.sql.*;

char maker, model[4], type[10], color[6];
float speed;
int ram, hd, screen, price;
```

```

Class.forName("<drive name>");

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

/* ask user for manufacturer */

/* get PCs made by the manufacturer */

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT * FROM PC WHERE model IN (" +
        "SELECT model FROM Product " +
        "WHERE maker = ? AND " +
        "type = 'pc'");
execStat.setString(1, maker);

ResultSet pcs = execStat.executeQuery();

while (pcs.next()) {

    model = pcs.getString(1);
    speed = pcs.getFloat(2);
    ram = pcs.getInt(3);
    hd = pcs.getInt(4);
    price = pcs.getInt(5);

    /* print fetched info */

}

/* get Laptops made by the manufacturer */

PreparedStatement execStat2 = myCon.prepareStatement(
    "SELECT * FROM Laptop WHERE model IN (" +
        "SELECT model FROM Product " +
        "WHERE maker = ? AND " +
        "type = 'laptop'");

execStat.setString(1, maker);

ResultSet laptops = execStat2.executeQuery();

while (laptops.next()) {

    model = laptops.getString(1);
    speed = laptops.getFloat(2);
    ram = laptops.getInt(3);
    hd = laptops.getInt(4);
    screen = laptops.getInt(5);
    price = laptops.getInt(6);

    /* print fetched info */

}

```



```

/* get Printers made by the manufacturer */

PreparedStatement execStat3 =
    "SELECT * FROM Printer WHERE model IN (" +
        "SELECT model FROM Product " +
        "WHERE maker = ? AND " +
        "type = 'printer'");

execStat3.setString(1, maker);

ResultSet printers = execStat3.executeQuery();

while (printers.next()) {

    model = printers.getString(1);
    color = printers.getString(2);
    type = printers.getString(3);
    price = printers.getInt(4);

    /* print fetched info */

}

d)

import java.sql.*;

int total_budget, rest_budget, pc_price, printer_price;
char pc_model[4], printer_model[4], color[6];
float min_speed;

Class.forName("<drive name>");

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

/* ask user for budget & the minimum speed of pc */

/* get the cheapest PC of the minimum speed */
PreparedStatement execStat = myCon.prepareStatement(
    "SELECT model, price FROM PC
    WHERE speed >= ? AND price <= ?
    ORDER BY price");

execStat.setFloat(1, min_speed);
execStat.setInt(2, total_budget);

ResultSet rs = execStat.executeQuery();

pc_model = rs.getString(1);
pc_price = rs.getInt(2);

if (!rs.next()) {
    printf("no pc found within the budget\n");
}

```

```

else {
    printf("pc model: %s\n", pc_model);
}

/* get Printer within the budget */
rest_budget = total_budget - pc_price;
color = "true";

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT model, price FROM Printer
    WHERE price <= ? AND color = ?
    ORDER BY price");

execStat.setInt(1, rest_budget);
execStat.setString(2, color);

ResultSet rs = execStat.executeQuery();

print_model = rs.getString(1);
print_price = rs.getInt(2);

if (!rs.next()) {
    color = "false";

    execStat.setInt(1, rest_budget);
    execStat.setString(2, color);

    ResultSet rs = execStat.executeQuery();

    print_model = rs.getString(1);
    print_price = rs.getInt(2);

    if (!rs.next())
        printf("no printer found within the budget\n");
    else
        printf("printer model: %s\n", printer_model);
}
else {
    printf("printer model: %s\n", printer_model);
}

e)

import java.sql.*;

char pmodel[4], pmaker, ptype[6];
float pspeed;
int pram, phd, pscreen, pprice, count;

char maker, model[4], type[10], color[6];
float speed;
int ram, hd, screen, price;

Class.forName("<drive name>");

```

```

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

/* ask user for manufacturer, model, speed, RAM, hard-disk, */
/* & price of a new PC */

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT COUNT(*) FROM PC WHERE model = ?");
execStat.setString(1, pmodel);

ResultSet rs = execStat.executeQuery();

Count = rs.getInt(1);

if (count > 0) {
    Printf("Warnning: The PC model already exists\n");
}
else {

    PreparedStatement execStat2 = myCon.prepareStatement(
        "INSERT INTO Product VALUES(?, ?, ?)");

    execStat2.setString(1, pmaker);
    execStat2.setString(2, pmodel);
    execStat2.setString(3, ptype);
    execStat2.executeUpdate();

    PreparedStatement execStat3 = myCon.prepareStatement(
        "INSERT INTO PC VALUES(?, ?, ?, ?, ?)");

    execStat3.setString(1, pmodel);
    execStat3.setFloat(2, pspeed);
    execStat3.setInt(3, pram);
    execStat3.setInt(4, phd);
    execStat3.setInt(5, pprice);
    execStat3.executeUpdate();

}

```

9.6.2

a)

```

import java.sql.*;

char cclass[20], maxFirepowerClass[20];
float firepower, maxFirepower;
int cnumGuns, cbore;

Class.forName("<drive name>");

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

```

```

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT class, numGuns, bore FROM Classes");
ResultSet classrs = execStat.executeQuery();

if(!classrs.next())
    /* print message and exit */ ;

cclass = classrs.getString(1);
cnumGuns = classrs.getString(2);
cbore = classrs.getString(3);

maxFirepower = cnumGuns * (power (cbore, 3));
maxFirepowerClass = cclass;

while(classrs.next()) {

    cclass = classrs.getString(1);
    cnumGuns = classrs.getString(2);
    cbore = classrs.getString(3);

    firepower = cnumGuns * (power (cbore, 3));

    if( firepower > maxFirepower )
    {
        maxFirepower = firepower;
        maxFirepowerClass = cclass;
    }
}

/* print maxFirepowerClass */

b)

import java.sql.*;

char ibattle[20], ocountry[20];
int colInfo;

Class.forName("<drive name>");

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

PreparedStatement execStat = myCon.prepareStatement(
    "SELECT COUNTRY FROM Classes C WHERE C.class IN (
        SELECT S.class FROM Ships S WHERE S.name IN (
            SELECT ship FROM Outcomes WHERE battle = ?))");

execStat.setString(1, ibattle);
ResultSet classrs = execStat.executeQuery();

while(classrs.next()) {

    ocountry = classrs.getString(1);

```

```

        /* print ocountry */
    }

c)

import java.sql.*;

char iclass[20], itype[3], icountry[20], iship[20];
int inumGuns, ibore, idisplacement, ilaunched;

Class.forName("<drive name>");

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

/* ask user for a class and other info for Classes table */

PreparedStatement execStat = myCon.prepareStatement(
    "INSERT INTO Classes VALUES (?, ?, ?, ?, ?, ?)");

execStat.setString(1, iclass);
execStat.setString(2, itype);
execStat.setString(3, icountry);
execStat.setInt(4, inumGuns);
execStat.setInt(5, ibore);
execStat.setInt(6, idisplacement);

execStat.executeUpdate();

/* ask user for a ship and launched */

PreparedStatement execStat2 = myCon.prepareStatement(
    "INSERT INTO Ships VALUES (?, ?, ?)");

while(there_is_input)
{
    execStat2.setSting(1, iship);
    execStat2.setSting(2, iclass);
    execStat2.setSting(3, ilaunched);

    execStat2.executeUpdate();

    /* ask user for a ship and launched */
}

d)

import java.sql.*;

char bname[20], bdate[8], newbdate[8];
char sname[20], lyear[4], newlyear[4];

Class.forName("<drive name>");

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

```

```

PreparedStatement execStat = myCon.prepareStatement(
    "Select b.name, b.date, s.name, s.launches " +
    "FROM Battles b, Outcomes o, Ships s " +
    "WHERE b.name = o.battle AND " +
    "o.ship = s.name AND " +
    "YEAR(b.date) < s.launches ");

ResultSet rs = execStat.executeQuery();

while(rs.next()) {

    bname = rs.getString(1);
    bdate = rs.getString(2);
    sname = rs.getString(3);
    lyear = rs.getString(4);

    /* prompt user and ask if a change is needed */

    if(change_battle)
    {
        /* get a new battle date to newbdate */
        PreparedStatement execStat2 = myCon.prepareStatement(
            "UPDATE Battles SET date = ? WHERE name = ?");
        execStat2.setString(1, newbdate);
        execStat2.setString(2, bname);
        execStat2.executeUpdate();
    }

    if(change_ship)
    {
        /* get a new launched year to newyear */
        PreparedStatement execStat3 = myCon.prepareStatement(
            "UPDATE Ships SET launched = ? WHERE name= ?");

        execStat3.setString(1, newyear);
        execStat3.setString(2, sname);
        execStat3.executeUpdate();
    }
}

```

9.7.1

```
a)
include(DB.php);

$myCon = DB::connect(<vendor>://<username>:<password>
                    <hostname>/<databasename>);

/* ask user for target price and read the answer into variable
   targetPrice */

$pcs = $myCon->query("SELECT model, price, speed FROM PC");

while($tuple = $pcs->fetchRow()) {

    $tempModel = $tuple[0];
    $tempPrice = $tuple[1];
    $tempSpeed = $tuple[2];

    if( /* the 1st fetch or tempPrice closer to targetPrice */ ) {
        $modelOfClosest = $tempModel;
        $priceOfClosest = $tempPrice;
        $speedOfClosest = $tempSpeed;
    }

}

/* Now, modelOfClosest is the model whose price is closest to
   target. We must get its manufacturer with a single-row select
   */

if (priceOfClosest == NULL ) /* no data fetched */
    /* print error message and exit */

$prepQuery = $myCon->prepare(
    "SELECT maker FROM Product WHERE model = ?");
$makers = $myCon->execute($prepQuery, $priceOfClosest);

/* print manf */

b)
include(DB.php);

$myCon = DB::connect(<vendor>://<username>:<password>
                    <hostname>/<databasename>);

/* ask user for minimum speed, ram, hd size, and screen size & get into
   args array */

$prepQuery = $myCon->prepare(
    "SELECT model, speed, ram, hd, screen, price, maker " .
    "FROM Laptop l, Product p " .
    "WHERE speed >= ? AND " .
    "       ram >= ? AND " .
    "       hd >= ? AND " .
    "       screen >= ? AND " .
    "l.model = p.model");
```

```

$products = $myCon->execute($prepQuery, $args);

while($tuple = $products->fetchRow()) {

    $model = $tuple[0];
    $speed = $tuple[1];
    $ram = $tuple[2];
    $hd = $tuple[3];
    $screen = $tuple[4];
    $price = $tuple[5];
    $maker = $tuple[6];

    /* print fetched info */

}

c)
include(DB.php);

$myCon = DB::connect(<vendor>://<username>:<password>
                    <hostname>/<databasename>);

/* ask user for manufacturer */

/* get PCs made by the manufacturer */

$prepQuery1 = $myCon->prepare(
    "SELECT * FROM PC WHERE model IN (" .
        "SELECT model FROM Product " .
        "WHERE maker = ? AND " .
        "type = 'pc'");
$pcs = $myCon->execute($prepQuery1, $maker);

while ($tuple = $pcs->fetchRow()) {

    $model = $tuple[0];
    $speed = $tuple[1];
    $ram = $tuple[2];
    $hd = $tuple[3];
    $price = $tuple[4];

    /* print fetched info */

}

/* get Laptops made by the manufacturer */

$prepQuery2 = $myCon->prepare(
    "SELECT * FROM Laptop WHERE model IN (" +
        "SELECT model FROM Product " +
        "WHERE maker = ? AND " +
        "type = 'laptop'");

$laptops = $myCon->execute($prepQuery2, $maker);

```



```

while ($tuple = $laptops->fetchRow()) {

    $model = $tuple[0];
    $speed = $tuple[1];
    $ram = $tuple[2];
    $hd = $tuple[3];
    $screen = $tuple[4];
    $price = $tuple[5];

    /* print fetched info */
}

/* get Printers made by the manufacturer */

$prepQuery3 = $myCon->prepare(
    "SELECT * FROM Printer WHERE model IN (" +
        "SELECT model FROM Product " +
        "WHERE maker = ? AND " +
        "type = 'printer'");

$printers = $myCon->execute($prepQuery3, $maker);

while ($tuple = $printers->fetchRow()) {

    $model = $tuple[0];
    $color = $tuple[1];
    $type = $tuple[2];
    $price = $tuple[3];

    /* print fetched info */

}

d)

include(DB.php);

$myCon = DB::connect(<vendor>://<username>:<password>
    <hostname>/<databasename>);

/* ask user for budget & the minimum speed of pc */

/* get the cheapest PC of the minimum speed */
$prepQuery1 = $myCon->prepare(
    "SELECT model, price FROM PC
        WHERE speed >= ? AND price <= ?
        ORDER BY price");

$pcs = $myCon->execute($prepQuery1, $args1);
/* $args1 - min_speed, total_budget */

if ($tuple = $pcs->fetchRow()) {

    $pc_model = $tuple[0];

```

```

    $pc_price = $tuple[1];

    /* print fetched info */

}
else
    /* print no pc found within the budget message */

/* get Printer within the budget */
$rest_budget = $total_budget - $pc_price;
$color = "true";

$prepQuery2 = $myCon.prepare(
    "SELECT model, price FROM Printer
    WHERE price <= ? AND color = ?
    ORDER BY price");

$printers = $myCon.execute($prepQuery1, $args2);
    /* $args2 - rest_budget, color */

if ($tuple = $pcs->fetchRow()) {

    $printer_model = $tuple[0];
    $printer_price = $tuple[1];

    /* print fetched info */

}
else {

    $color = "false"
    $printers = $myCon.execute($prepQuery1, $args2);
        /* $args2 - rest_budget, color */

    if ($tuple = $pcs->fetchRow()) {

        $printer_model = $tuple[0];
        $printer_price = $tuple[1];

        /* print fetched info */
    }
    else
        /* print no printer found within the budget message */
    }

e)
include(DB.php);

$myCon = DB::connect(<vendor>:://<username>:<password>
    <hostname>/<databasename>);

/* ask user for manufacturer, model, speed, RAM, hard-disk, */
/* & price of a new PC */

$prepQuery1 = $myCon.prepare(

```

```

        "SELECT COUNT(*) FROM PC WHERE model = ?");

$rs = $myCon.execute($prepQuery1, $pmodel);
$tuple = $rs->fetchRow();
$count = $tuple[0];

if ($count > 0) {
    Printf("Warnning: The PC model already exists\n");
}
else {

    $prepStmt2 = $myCon.prepare(
        "INSERT INTO Product VALUES(?, ?, ?)");
    /* pmaker, pmode, & ptype are $args1 */

    $result = $myCon->execute($prepStmt2, $args1);

    $prepStmt3 = $myCon->prepare(
        "INSERT INTO PC VALUES(?, ?, ?, ?, ?)");

    /* pmodel, pspeed, pram, phd, & pprice are in are $args2 */

    $result = $myCon->execute($prepStmt3, $args2);

}

```

9.7.2

a)

```

include(DB.php);

$myCon = DB::connect(<vendor>://<username>:<password>
    <hostname>/<databasename>);

$classrs = $myCon->query(
    "SELECT class, numGuns, bore FROM Classes");
$tuple = $classrs->fetchRow();

if(!$tuple)
    /* print message and exit */ ;

$class = $tuple[0];
$numGuns = $tuple[1];
$bore = $tuple[2];

$maxFirepower = $numGuns * ($bore * $bore * $bore);
$maxFirepowerClass = $class;

while($tuple = $classrs->fetchRow()) {

    $class = $tuple[0];
    $numGuns = $tuple[1];
    $bore = $tuple[2];

    $firepower = $numGuns * ($bore * $bore * $bore);

```

```

    if( $firepower > $maxFirepower )
    {
        $maxFirepower = $firepower;
        $maxFirepowerClass = $cclass;
    }
}

/* print maxFirepowerClass */

b)

include(DB.php);

$myCon = DB::connect(<vendor>:://<username>:<password>
                    <hostname>/<databasename>);

$prepQuery = $myCon->prepare(
    "SELECT COUNTRY FROM Classes C WHERE C.class IN ( " .
        "SELECT S.class FROM Ships S WHERE S.name IN ( " .
            "SELECT ship FROM Outcomes WHERE battle = ?))");
/* battle in $ibattle */

$classrs = $myCon->execute($prepQuery, $ibattle);

while($tuple = $classrs->fetchRow()) {

    $ocountry = $tuple[0];

    /* print ocountry */
}

c)

include(DB.php);

$myCon = DB::connect(<vendor>:://<username>:<password>
                    <hostname>/<databasename>);

/* ask user for a class and other info for Classes table */

$prepStmt1 = $myCon->prepare(
    "INSERT INTO Classes VALUES (?, ?, ?, ?, ?, ?)");

/* $iclass, $itype, $icountry, $inumGuns, $ibore, & $idisplacement in
$args1 */

$result = $myCon->execute($prepStmt1, $args1);

/* ask user for a ship and launched */

$prepStmt2 = $myCon->prepare(
    "INSERT INTO Ships VALUES (?, ?, ?)");

/* $iship, $iclass, $ilaunched in $args2 */

```

```

while(there_is_input)
{
    $result = $myCon->execute($prepStmt2, $args2);

    /* ask user for a ship and launched & get into args2*/
}

d)

include(DB.php);

$myCon = DB::connect(<vendor>:://<username>:<password>
                    <hostname>/<databasename>);

$rs = $myCon->query(
    "SELECT b.name, b.date, s.name, s.launched " .
    "FROM Battles b, Outcomes o, Ships s " .
    "WHERE b.name = o.battle AND " .
    "o.ship = s.name AND " .
    "YEAR(b.date) > s.launched ");

while($tuple = $rs.fetchRow()) {

    $bname = $tuple[0];
    $bdate = $tuple[1];
    $sname = $tuple[2];
    $lyear = $tuple[3];

    /* prompt user and ask if a change is needed */

    if(change_battle)
    {
        /* get a new battle date to newbdate in $args2*/
        $prepStmt2 = $myCon->prepare(
            "UPDATE Battles SET date = ? WHERE name = ?");
        $result = $myCon->execute($prepStmt2, $args2);
    }

    if(change_ship)
    {
        /* get a new launched year to newlyear in $args3 */
        $prepStmt3 = $myCon->prepare(
            "UPDATE Ships SET launched = ? WHERE name= ?");

        $result = $myCon->execute($prepStmt3, $args3);
    }
}

```