Database Application Development Assignment

Part 1

Objective:

In this assignment, you create a simple HR application using the C++ programming language and Oracle. This assignment helps students learn a basic understanding of application development using C++ programming and an Oracle database.

Submission:

Your submission will be a single text-based .cpp file including your C++ program for the Database Application assignment.

AS_P1_LASTNAME.cpp

Your submission needs to be commented.

Instruction:

In this assignment, we use the same database that you use for the labs.

Note: For each query in your assignment, make sure you handle the errors and display the proper message including the error_code and the error message.

```
try{
        ...
}
catch (SQLException& sqlExcp) {
     cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
}
```

Connecting to an Oracle database from a C++ Program

In your function ***main()***, create a connection to your database.

First, declare the environment and the connection variables.

```
Environment* env = nullptr;
Connection* conn = nullptr;
```

Define and initialize the variable to store the username, password, and the host address.

```
string user = "username";
string pass = "password";
string constr = "myoracle12c.senecacollege.ca:1521/oracle12c";
```

Use the same Oracle username and password that you use for your labs and assignments.
Create the environment and the connection. Make sure you handle any errors may be thrown as you program is executed.

```
env = Environment::createEnvironment(Environment::DEFAULT);
conn = env->createConnection(user, pass, constr);
```

Remember to terminate and close the connection and the environment, when your program terminates.

```
env->terminateConnection(conn);
Environment::terminateEnvironment(env);
```

You will implement the following functions:

int menu(void);

The ***menu()*** function returns an integer value which is the selected option by the user from the menu. This function displays the following menu options:

1) Find Employee
2) Employees Report
3) Add Employee
4) Update Employee
5) Remove Employee
0) Exit

Before printing the menu, display the following title on the screen

```
********************* HR Menu *********************
```

Prompt the user to enter an option (0 to 5). If the user enters an incorrect option, the user is asked to enter an option again. When the user enters a correct option (0 to 5), the function returns the selected value.

If the user selects 0 (Exit), the program terminates.

Every time you call this function, the menu is displayed and the user is prompted to enter a value between 0 and 5. The function keeps asking the user enter a value until the user enters a correct number.

int findEmployee(Connection *conn, int employeeNumber, struct Employee *emp);

This function receives an OCCI pointer (a reference variable to an Oracle database), an integer number as the employee number, and a pointer to a variable of type Employee. The function returns 0 if the employee does not exist. It returns 1 if the employee exits.

To store the employee data in the ***findEmployee()*** function, we use a pointer that refers to a variable of type structure called Employee. The Employee structure has the following members:

```
struct Employee{
     int employeeNumber;
     char lastName[50];
     char firstName[50];
     char extension[10];
     char email[100];
     char officecode[10];
     int  reportsTo[100];
     char jobTitle[50];
};
```

The *ReportsTo* member stores the employee ID of the employee who is the manager of the given employee number.
If the employee exists, store the employee data into the members of an Employee variable using the third parameter in the ***findEmployee()*** function which references to that variable of type Employee.

void displayEmployee(Connection *conn, struct Employee emp);

If the user selects option 1, prompt the user to enter a value for the employee number. Then, call function ***findEmployee()*** to check if the employee with the given employee number exists. If the returning value of function ***findEmployee()*** is 0, display a proper error message.
Sample error message:

Employee 1122 does not exist.

Otherwise, call the function ***displayEmployee()*** to display the employee information.
This function receives a Connection pointer (a reference variable to an Oracle database) and the members of a variable of type Employee and displays all members of the *emp* parameter.

See the following sample output:

```
******************** HR Menu ********************
1) Find Employee
2) Employees Report
3) Add Employee
4) Update Employee
5) Remove Employee
0) Exit
Enter an option (0-5): 1
Enter Employee Number: 1002

-------------- Employee Information -------------
Employee Number: 1002
Last Name: Murphy
First Name: Diane
Extension: x5800
```

```
      Email: dmurphy@classicmodelcars.com
      Office Code: 1
      Manager ID: 0
      Job Title: President

      ********************HR Menu********************
      1)      Find Employee
      2)      Employees Report
      3)      Add Employee
      4)      Update Employee
      5)      Remove Employee
      0)      Exit
      Enter an option (0-5): 1
      Enter Employee Number: 1078
      Employee 1078 does not exist.
```

void displayAllEmployees(Connection *conn);

If the user selects option 2 (Employees Report), call function *displayAllEmployees().*
This function receives a pointer of type OCCI Conection (a reference variable to an Oracle database) and displays all employees' information if exist.

```
********************HR Menu********************

      1) Find Employee
      2) Employees Report
      3) Add Employee
      4) Update Employee
      5) Remove Employee
      0) Exit
      Enter an option (0-5): 2

------  ---------------  ---------------------------------  ----------------  ---------  -----------------
ID      Employee Name    Email                              Phone             Extension  Manager Name
-----   ---------------  ---------------------------------  ----------------  ---------  -----------------
1002    Diane Murphy     dmurphy@classicmodelcars.com       +1 650 219 4782   x5800
1056    Mary Patterson   mpatterso@classicmodelcars.com     +1 650 219 4782   x4611      Diane Murphy
1076    Jeff Firrelli    jfirrelli@classicmodelcars.com     +1 650 219 4782   x9273      Diane Murphy
1088    William Patterson wpatterson@classicmodelcars.com   +61 2 9264 2451   x4871      Mary Patterson
1102    Gerard Bondur    gbondur@classicmodelcars.com       +33 14 723 4404   x5408      Mary Patterson
1143    Anthony Bow      abow@classicmodelcars.com          +1 650 219 4782   x5428      Mary Patterson
1165    Leslie Jennings  ljennings@classicmodelcars.com     +1 650 219 4782   x3291      Anthony Bow
1166    Leslie Thompson  lthompson@classicmodelcars.com     +1 650 219 4782   x4065      Anthony Bow
1188    Julie Firrelli   jfirrelli@classicmodelcars.com     +1 215 837 0825   x2173      Anthony Bow
1216    Steve Patterson  spatterson@classicmodelcars.com    +1 215 837 0825   x4334      Anthony Bow
1286    Foon Yue Tseng   ftseng@classicmodelcars.com        +1 212 555 3000   x228       Anthony Bow
1323    George Vanauf    gvanauf@classicmodelcars.com       +1 212 555 3000   x4102      Anthony Bow
1337    Loui Bondur      lbondur@classicmodelcars.com       +33 14 723 4404   x6493      Gerard Bondur
1370    Gerard Hernandez ghernande@classicmodelcars.com     +33 14 723 4404   x2028      Gerard Bondur
1401    Pamela Castillo  pcastillo@classicmodelcars.com     +33 14 723 4404   x2759      Gerard Bondur
1501    Larry Bott       lbott@classicmodelcars.com         +44 20 7877 2041  x2311      Gerard Bondur
1504    Barry Jones      bjones@classicmodelcars.com        +44 20 7877 2041  x102       Gerard Bondur
1611    Andy Fixter      afixter@classicmodelcars.com       +61 2 9264 2451   x101       William Patterson
1612    Peter Marsh      pmarsh@classicmodelcars.com        +61 2 9264 2451   x102       William Patterson
1619    Tom King         tking@classicmodelcars.com         +61 2 9264 2451   x103       William Patterson
1621    Mami Nishi       mnishi@classicmodelcars.com        +81 33 224 5000   x101       Mary Patterson
1625    Yoshimi Kato     ykato@classicmodelcars.com         +81 33 224 5000   x102       Mami Nishi
1702    Martin Gerard    mgerard@classicmodelcars.com       +33 14 723 4404   x2312      Gerard Bondur
```

<u>Note</u>: For this report, you may need to query more than one table (join).

If the query does not return any rows, display a proper message:

There is no employees’ information to be displayed.