```cpp
//=============================================
// Names: Mayuran Selvarasa, Md Rafi Al Arabi Bhuiyan and Mohammad Yeamin Khan
// Student Number: 019126143,147307193,114964190
// Email:          mselvarasa1@myseneca.ca , mraabhuiyan@myseneca.ca , mykhan10@myseneca.ca
// Section:        DBS211NFF
// Workshop:       Part 2 of Assignment
//=============================================

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <occi.h>
#include <iomanip>
#include <sstream>
#include <string>

#include "Menu.h"

using namespace std;

namespace dbs
{
	string user = "dbs211_202f21";
	string pass = "22523592";
	string constr = "myoracle12c.senecacollege.ca:1521/oracle12c";
	string tablename = "EMPLOYEES";

	void menuTitles()
	{
		cout << "*********************HR Menu*********************" << endl;
		cout << "1) Find Employee\n";

		cout << "2) Employees Report\n";

		cout << "3) Add Employee\n";

		cout << "4) Update Employee\n";

		cout << "5) Remove Employee\n";

		cout << "0) Exit\n";
	}

	int checkValue()
	{
		string str = "\0";
		int value = -1;

		do
		{
			cin >> str;
			cout << "Choice is " << str << endl;

			if ((str.compare("1") == 0) || (str.compare("2") == 0)
				|| (str.compare("3") == 0) || (str.compare("4") == 0)
				|| (str.compare("5") == 0) || (str.compare("0") == 0))
			{
				stringstream ss(str);
				ss >> value;
			}

			else
			{
				cout << "Invalid choice, only numbers from 0 to 5 are acceptable, retry: ";
				value = -1;
			}

		} while (value == -1);
```

```
                return value;
        }

        int menu(void)
        {
                int value = -1;
                int empNum = 0;

                menuTitles();
                cout << "Enter an option (0-5): ";

                value = checkValue();

                return value;
        }

        void displayEmployee(Employee* emp)
        {
                cout << "\nEmployee Number: " << emp->employeeNumber << endl;
                cout << "Last Name: " << emp->lastName << endl;
                cout << "First Name: " << emp->firstName << endl;
                cout << "Extension: " << emp->extension << endl;
                cout << "Email: " << emp->email << endl;
                cout << "Office Code: " << emp->officeCode << endl;
                cout << "Manager ID: " << emp->managerID << endl;
                cout << "Job Title: " << emp->jobTitle << endl;
                cout << endl;
        }
        void displayAllEmployees()
        {
                //Defining Objects
                Environment* env = Environment::createEnvironment(Environment::DEFAULT);
                Connection* conn = env->createConnection(user, pass, constr);

                string query = "SELECT  EMPLOYEENUMBER, FIRSTNAME || ' ' || LASTNAME AS empName, EXTENSION, EMAIL,
OFFICECODE, REPORTSTO, JOBTITLE FROM " + tablename + " ORDER BY EMPLOYEENUMBER";

                Statement* stmt = conn->createStatement(query);
                ResultSet* rs = stmt->executeQuery();

                const char separator = '  ';
                const int width1 = 15;
                const int width2 = 20;
                const int width3 = 35;

                cout << "Displaying Employee Report" << endl;

                cout << left << setw(width1) << setfill(separator) << "\nID";
                cout << left << setw(width2) << setfill(separator) << " Employee Name";
                cout << left << setw(width1) << setfill(separator) << " Extension";
                cout << left << setw(width3) << setfill(separator) << " Email";
                cout << left << setw(width2) << setfill(separator) << " Office Code";
                cout << left << setw(width1) << setfill(separator) << " Manager ID";
                cout << left << setw(width1) << setfill(separator) << " Job Title";
                char oldFill = cout.fill('-');
                cout.width(135);
                cout << "";
                cout.fill('-');
                cout << endl;

                if (!rs->next()) {
                        // if the result set is empty
                        cout << "There is no employees� information to be displayed." << endl;
                }
                else
                {
                        do
                        {
                                       cout << left << setw(width1) << setfill(separator) << rs->getInt(1);
                                       cout << left << setw(width2) << setfill(separator) << rs->getString(2);
                                       cout << left << setw(width3) << setfill(separator) << rs->getString(3);
                                       cout << left << setw(width2) << setfill(separator) << rs->getString(4);
                                       cout << left << setw(width1) << setfill(separator) << rs->getInt(5);
                                       cout << left << setw(width1) << setfill(separator) << rs->getInt(6);
                                       cout << left << setw(width1) << rs->getString(7) << endl;
                                       cout << endl;

                        } while (rs->next());
                }

                conn->terminateStatement(stmt);
        }
        void displayFindEmployee() {
                int employeeNumber = 0;

                cout << "Employee Number:\n";
                cin >> employeeNumber;

                Employee* emp = new Employee();
                Environment* env = Environment::createEnvironment(Environment::DEFAULT);
                Connection* conn = env->createConnection(user, pass, constr);

                if (findEmployee(conn, employeeNumber, emp) <= 0)
                {
                        cout << "The employee with ID " << employeeNumber << " does not exist." << endl;
                }
                terminate(env, conn);

        }
        int findEmployee(Connection* conn, int employeeNumber, Employee* emp)
        {
                string query = "SELECT EMPLOYEENUMBER, LASTNAME, FIRSTNAME, EXTENSION, EMAIL, OFFICECODE, REPORTSTO,
JOBTITLE FROM " + tablename + " WHERE EMPLOYEENUMBER= :c1";

                Statement* stmt = conn->createStatement(query);
                stmt->setInt(1, employeeNumber);
                ResultSet* rs = stmt->executeQuery();

                int rsi = 0;

                if (rs->next()) {
                        do {
                                       int a = rs->getInt(1);

                                       if (a == employeeNumber)
                                       {
                                               emp->employeeNumber = employeeNumber;
                                               strcpy(emp->lastName, rs->getString(2).c_str());
                                               strcpy(emp->firstName, rs->getString(3).c_str());
                                               strcpy(emp->extension, rs->getString(4).c_str());
                                               strcpy(emp->email, rs->getString(5).c_str());
                                               emp->officeCode = rs->getInt(6);
                                               emp->managerID = rs->getInt(7);
                                               strcpy(emp->jobTitle, rs->getString(8).c_str());

                                               displayEmployee(emp);
```

```
                                                rsi = 1;
                                    }
                        } while (rs->next());

            }
            cout << endl;

            conn->terminateStatement(stmt);

            return rsi;
}
void displayAddEmployee() {
            Employee* emp = new Employee();

            cout << "-------------- New Employee Information -------------" << endl;
            cout << "Employee Number:\n";
            cin >> emp->employeeNumber;

            cout << "Last Name:\n";
            cin >> emp->lastName;

            cout << "First Name:\n";
            cin >> emp->firstName;

            cout << "Extension:\n";
            cin >> emp->extension;

            cout << "Email:\n";
            cin >> emp->email;

            cout << "Office Code:\n";
            cin >> emp->officeCode;

            cout << "Manager ID:\n";
            cin >> emp->managerID;

            cout << "Job Title:\n";
            cin >> emp->jobTitle;

            insertEmployee(emp);
}
void insertEmployee(struct Employee* emp) {
            Environment* env = Environment::createEnvironment(Environment::DEFAULT);
            Connection* conn = env->createConnection(user, pass, constr);

            int checkEmployee = findEmployee(conn, emp->employeeNumber, emp);
            if (checkEmployee <= 0)
            {
                        insertEmployee(conn, emp);
                        cout << "The new employee is added successfully\n";
            }
            else {
                        cout << "An employee with the same employee number exists.\n";
            }

            terminate(env, conn);
}
void insertEmployee(Connection* conn, struct Employee* emp) {
            string query = "INSERT INTO " + tablename + " values (:c1, :c2, :c3, :c4,:c5, :c6, :c7, :c8)";
            Statement* stmt = conn->createStatement("INSERT INTO " + tablename + " values (:c1, :c2, :c3, :c4,:c5,
:c6, :c7, :c8)");
            stmt->setInt(1, emp->employeeNumber);
            stmt->setString(2, emp->lastName);
            stmt->setString(3, emp->firstName);
            stmt->setString(4, emp->extension);
            stmt->setString(5, emp->email);
            stmt->setInt(6, emp->officeCode);
            stmt->setInt(7, emp->managerID);
            stmt->setString(8, emp->jobTitle);
            stmt->executeUpdate();

            conn->commit();
            conn->terminateStatement(stmt);
}
void displayUpdateEmployee() {
            int employeeNumber = 0;

            cout << "Employee Number:\n";
            cin >> employeeNumber;

            updateEmployee(employeeNumber);
}
void updateEmployee(int employeeNumber) {
            Employee* emp = new Employee();
            Environment* env = Environment::createEnvironment(Environment::DEFAULT);
            Connection* conn = env->createConnection(user, pass, constr);

            int checkEmployee = findEmployee(conn, employeeNumber, emp);

            if (checkEmployee > 0)
            {
                        cout << "Last Name:";
                        cout << emp->lastName << endl;
                        cout << "First Name:";
                        cout << emp->firstName << endl;
                        cout << "Extension:\n";
                        cin >> emp->extension;
                        updateEmployee(conn, emp);
                        cout << "The employee's extension is updated successfully." << endl;
            }
            else {
                        cout << "The employee with ID " << employeeNumber << " does not exist." << endl;
            }
            terminate(env, conn);
}
void updateEmployee(Connection* conn, Employee* emp) {
            string query = "UPDATE " + tablename + " SET EXTENSION =:c1 WHERE EMPLOYEENUMBER=:c2";
            Statement* stmt = conn->createStatement(query);
            stmt->setString(1, emp->extension);
            stmt->setInt(2, emp->employeeNumber);
            stmt->executeUpdate();

            conn->commit();
            conn->terminateStatement(stmt);
}
void displayDeleteEmployee() {
            int employeeNumber = 0;
            cout << "Employee Number:\n";
            cin >> employeeNumber;

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

            deleteEmployee(conn, employeeNumber);

            terminate(env, conn);
```

```cpp
            cout << "" << endl;
        }
        void deleteEmployee(Connection* conn, int employeeNumber) {
            Employee* emp = new Employee();

            int checkEmployee = findEmployee(conn, employeeNumber, emp);
            if (checkEmployee > 0)
            {
                string query = "DELETE FROM " + tablename + " WHERE EMPLOYEENUMBER=:c1";
                Statement* stmt = conn->createStatement(query);
                stmt->setInt(1, employeeNumber);
                stmt->execute();

                conn->commit();
                conn->terminateStatement(stmt);
                cout << "The employee with ID " << employeeNumber << " is deleted successfully." << endl;
            }
            else {
                cout << "The employee with ID " << employeeNumber << " does not exist." << endl;
            }

        }
        void terminate(Environment* env, Connection* conn)
        {
            env->terminateConnection(conn);
            Environment::terminateEnvironment(env);
        }
    }
```