```cpp
#include <iostream>
#include <string>
#include <occi.h>
#include <cctype>

using oracle::occi::Environment;
using oracle::occi::Connection;
using namespace oracle::occi;
using namespace std;


// basic struct to hold product details for the cart
struct ShoppingCart {

    int product_id;

    double price;

    int quantity;
};

// this is the main menu shown at program start
// gives options to login or quit
int mainMenu() {
    int option = 0;
    do {
        cout << "\n******************** Main Menu ********************\n"
            << "1)\tLogin\n"

            << "0)\tExit\n";

        if (option != 0 && option != 1) {

            cout << "You entered a wrong value. Enter an option (0-1): ";
        }

        else {

            cout << "Enter an option (0-1): ";
        }

        cin >> option;

    } while (option != 0 && option != 1);

    return option;
}

// then this is the sub-menu after login
// lets user place/check/cancel orders
int subMenu() {
    int option = -1;
    do {
        cout << "\n******************** Customer Service Menu ********************\n";

        cout << "1) Place an order\n";

        cout << "2) Check an order status\n";

        cout << "3) Cancel an order\n";

        cout << "0) Exit\n";

        cout << "Enter an option (0-3): ";

        cin >> option;

        if (option < 0 || option > 3)

            cout << "Invalid option. Try again..." << endl;
    } while (option < 0 || option > 3);

    return option;
}

// thsi calls stored procedure to get product price from DB
// returns price if found, otherwise 0
double findProduct(Connection* conn, int product_id) {

    Statement* stmt = conn->createStatement();

    stmt->setSQL("BEGIN find_product(:1, :2); END;");

    double price;

    stmt->setInt(1, product_id);
```

```
    stmt->registerOutParam(2, Type::OCCIDOUBLE, sizeof(price));

    stmt->executeUpdate();

    price = stmt->getDouble(2);

    conn->terminateStatement(stmt);
    return price > 0 ? price : 0;
}

// this shows what's in the cart and calculates the total price
void displayProducts(struct ShoppingCart cart[], int productCount) {

    if (productCount > 0) {

        double totalPrice = 0;

        cout << "------- Ordered Products ---------" << endl;

        for (int i = 0; i < productCount; ++i) {

            cout << "---Item " << i + 1 << endl;

            cout << "Product ID: " << cart[i].product_id << endl;

            cout << "Price: " << cart[i].price << endl;

            cout << "Quantity: " << cart[i].quantity << endl;

            totalPrice += cart[i].price * cart[i].quantity;
        }
        cout << "---------------------------------\nTotal: " << totalPrice << endl;
    }
}

// this checks if the customer exists in the DB using stored proc
int customerLogin(Connection* conn, int customerId) {
    Statement* stmt = conn->createStatement();

    stmt->setSQL("BEGIN find_customer(:1, :2); END;");

    int found;

    stmt->setInt(1, customerId);

    stmt->registerOutParam(2, Type::OCCIINT, sizeof(found));

    stmt->executeUpdate();

    found = stmt->getInt(2);

    conn->terminateStatement(stmt);

    return found;
}

// this allows the user to add up to 5 items to the cart
// so it fetches price and verifies product exists
int addToCart(Connection* conn, struct ShoppingCart cart[]) {

    cout << "-------------- Add Products to Cart --------------" << endl;

    for (int i = 0; i < 5; ++i) {

        int productId;

        int qty;

        ShoppingCart item;

        int choice;

        do {
            cout << "Enter the product ID: ";

            cin >> productId;

            if (findProduct(conn, productId) == 0) {

                cout << "The product does not exist. Try again..." << endl;
            }
        } while (findProduct(conn, productId) == 0);

        cout << "Product Price: " << findProduct(conn, productId) << endl;

        cout << "Enter the product Quantity: ";
```

```
        cin >> qty;

        item.product_id = productId;

        item.price = findProduct(conn, productId);

        item.quantity = qty;

        cart[i] = item;

        if (i == 4)
            return i + 1;

        do {

            cout << "Enter 1 to add more products or 0 to check out: ";

            cin >> choice;

        } while (choice != 0 && choice != 1);

        if (choice == 0)

            return i + 1;
    }
}

// this finalizes the order and sends it to the DB
// so it calls stored procs to create order and add order items
int checkout(Connection* conn, struct ShoppingCart cart[], int customerId, int productCount) {
    char choice;
    do
    {
    cout << "Would you like to checkout ? (Y / y or N / n) ";

        cin >> choice;

        if (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n')

            cout << "Wrong input. Try again..." << endl;

    } while (choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n');


    if (choice == 'N' || choice == 'n')
    {
        cout << "The order is cancelled." << endl;
        return 0;

    }

    else {
        Statement* stmt = conn->createStatement();
        stmt->setSQL("BEGIN add_order(:1, :2); END;");

        int next_order_id;

        stmt->setInt(1, customerId);

        stmt->registerOutParam(2, Type::OCCIINT, sizeof(next_order_id));

        stmt->executeUpdate();

        next_order_id = stmt->getInt(2);

        for (int i = 0; i < productCount; ++i) {
            stmt->setSQL("BEGIN add_order_item(:1, :2, :3, :4, :5); END;");

            stmt->setInt(1, next_order_id);

            stmt->setInt(2, i + 1);

            stmt->setInt(3, cart[i].product_id);

            stmt->setInt(4, cart[i].quantity);

            stmt->setDouble(5, cart[i].price);

            stmt->executeUpdate();
        }

        cout << "The order is successfully completed." << endl;
        conn->terminateStatement(stmt);
        return 1;
    }
}
```

```
// this one checks the status of an order from the DB
void displayOrderStatus(Connection* conn, int orderId, int customerId) {

    Statement* stmt = conn->createStatement();

    try {
        stmt->setSQL("BEGIN customer_order(:1, :2); END;");

        stmt->setInt(1, customerId);

        stmt->setInt(2, orderId);

        stmt->registerOutParam(2, Type::OCCIINT, sizeof(orderId));

        stmt->executeUpdate();

        int verifiedOrderId = stmt->getInt(2);

        if (verifiedOrderId == 0) {
            cout << "Order ID is not valid." << endl;
        }

        else {
            stmt->setSQL("BEGIN display_order_status(:1, :2); END;");

            stmt->setInt(1, orderId);

            stmt->registerOutParam(2, Type::OCCISTRING, 30);

            stmt->executeUpdate();

            string status = stmt->getString(2);
            if (status.empty()) {
                cout << "Order does not exist." << endl;
            }
            else {
                cout << "Order is " << status << "." << endl;
            }
        }
    }
    catch (SQLException& sqlExcp) {
        cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage() << endl;
    }

    conn->terminateStatement(stmt);
}

// this cancels an order by calling a stored proc
// so it checks if the order is valid before trying to cancel
void cancelOrder(Connection* conn, int orderId, int customerId) {
    Statement* stmt = conn->createStatement();
    try {
        stmt->setSQL("BEGIN customer_order(:1, :2); END;");

        stmt->setInt(1, customerId);

        stmt->setInt(2, orderId);

        stmt->registerOutParam(2, Type::OCCIINT, sizeof(orderId));

        stmt->executeUpdate();

        int verifiedOrderId = stmt->getInt(2);

        if (verifiedOrderId == 0) {
            cout << "Order ID is not valid." << endl;
        }

        else {
            stmt->setSQL("BEGIN cancel_order(:1, :2); END;");

            stmt->setInt(1, orderId);

            stmt->registerOutParam(2, Type::OCCIINT, sizeof(int));

            stmt->executeUpdate();

            int cancelFlag = stmt->getInt(2);

            if (cancelFlag == 1)
                cout << "Order is canceled." << endl;
            else
                cout << "Order does not exist." << endl;
        }
    }
    catch (SQLException& sqlExcp) {
```

```
        cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage() << endl;
    }
    conn->terminateStatement(stmt);
}

int main() {
    Environment* env = nullptr;

    Connection* conn = nullptr;

    string user = "dbs311_251nra22";

    string pass = "30027753";

    string constr = "myoracle12c.senecacollege.ca:1521/oracle12c";

    try {
        env = Environment::createEnvironment(Environment::DEFAULT);

        conn = env->createConnection(user, pass, constr);

        cout << "Connection is successful!" << endl;


        int choice;

        int customerId;

        do {
            choice = mainMenu();

            if (choice == 1) {

                cout << "Enter the customer ID: ";

                cin >> customerId;

                if (customerLogin(conn, customerId) == 0) {

                    cout << "The customer does not exist." << endl;

                }
                else {

                    ShoppingCart cart[5];

                    int productCnt = addToCart(conn, cart);

                    displayProducts(cart, productCnt);

                    checkout(conn, cart, customerId, productCnt);
                }

            }


        } while (choice != 0);

        cout << "Good bye!..." << endl;

        env->terminateConnection(conn);
        Environment::terminateEnvironment(env);
    }
    catch (SQLException& sqlExcp) {
        cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
    }
    return 0;
}
```