***OoP Semester project report***

**Submitted To :** Rizwan Ali

**Submitted By :** Muhammad Ali Turk

**Registration No. :** B21F0600DS026

**Section :** DS

**Department :** IT and CS

**Programming statement :**

Make a program using inheritance , operator overloading and exceptional handling .

**Objective** :

The objective of this program is to provide an interface to the user , Where the user can check the games of the gaming consoles available in the game shop and check their prices and give their grand total of the games .

**Our program :**

We have made a game shop system using C++(OOP) . In our program we took two gaming consoles . First one being Xbox and the Other one being Ps4 . Now when you start the program the user asks for whether u would like to buy a game of Ps4 or Xbox . When enter either it provides you with the multiple options of the game of the entered console . It shows the total number of games in the console and their avalibilty in stock and when you chose a certain game it provides you with its price and then it asks you whether you would like to buy any other game you either enter y for yes or n for no if yes it continues to ask for the console first and later the games till you chose to exit . And it gives you your grand total and the program ends in any case of exception in the bigenning it asks for your phone number .

**Working :**

This program is a simple game shop simulation where customers can choose between Xbox and PlayStation game consoles and select games to purchase. The program keeps track of the available stock for each game and calculates the total amount to be paid by the customer.

Let's go through the program step by step:

1. The program includes the necessary header files `iostream` and `exception` and defines the `std` namespace.

2. The `GameShop` class is defined, which serves as the base class for Xbox and PlayStation classes. It has member variables such as `station\_type` (to store the chosen console type), `ph` (to store the phone number of the customer), `n` (to store the game selection), and `pri` (to store the price of the selected game). It also includes static variables for tracking the stock of each game.

3. The `display()` function is defined within the `GameShop` class to display a welcome message to the game shop.

4. The `get()` function is defined within the `GameShop` class to get the console type choice from the user.

5. The `Amount()` function is defined within the `GameShop` class to display the price of the selected game.

6. The addition operator (`operator+`) is overloaded within the `GameShop` class to add the prices of two game selections.

7. Static variables `Taken`, `GtaV`, and `Fifa23` are initialized outside the class definition.

8. The `xbox` class is defined, which is derived from the `GameShop` base class. It includes a function `x()` to handle Xbox game selection and stock management.

9. The `p()` function is defined within the `PS` class, which is derived from the `GameShop` base class. It handles PlayStation game selection and stock management.

10. The `main()` function begins by declaring objects `obj1`, `obj2`, and `total` of type `GameShop`. It also displays the initial welcome message.

11. Inside the `do-while` loop, the user is prompted to enter the console type choice using the `get()` function. If the choice is Xbox, an object of the `xbox` class is created, and the `x()` function is called to handle Xbox game selection. If the choice is PlayStation, an object of the `PS` class is created, and the `p()` function is called to handle PlayStation game selection.

12. The `operator+` is used to add the current purchase (`obj3` or `obj4`) to the total purchase (`total`).

13. After each game selection, the user is prompted to check for other games by entering 'Y' or 'N'.

14. The loop continues until the user enters 'N' or 'n'.

15. After the loop, the program displays the total amount to be paid (`total.pri`) and a farewell message.

**Source code:**

**#include<iostream>**

**#include<exception>**

**using namespace std;**

**class GameShop**

**{**

**public:**

**string station\_type;**

**string ph;**

**int n, pri;**

**static int Taken,GtaV,Fifa23;**

**static int Gow,Rdp,Fortnite;**

**void display()**

**{**

**cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\n";**

**cout << "\t\t\t\t\tWelcome to HA games\n ";**

**cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\n";**

**}**

**void get()**

**{**

**cout << "Which station do you want to checkout (Xbox or PS):" << endl;**

**cout << "Enter 'Xbox' for Xbox or 'PS' for Play Station\n";**

**cin >> station\_type;**

**}**

**void Amount(int price)**

**{**

**pri = price;**

**cout << station\_type << " Price: Rs. " << pri << endl;**

**}**

**// Overloading the addition operator**

**GameShop operator+(const GameShop& other)**

**{**

**GameShop result;**

**result.pri = this->pri + other.pri;**

**return result;**

**}**

**};**

**int GameShop::Taken=2;**

**int GameShop::GtaV=2;**

**int GameShop::Fifa23=2;**

**class xbox : public GameShop**

**{**

**public:**

**void x()**

**{**

**cout << "We have the following games on Xbox. Please choose one: " << endl;**

**cout << "1. Takken 7 Stock: "<<Taken<<"\n2. GTA V Stock: "<<GtaV<<"\n3. FIFA 23 Stock: "<<Fifa23<<"\n4. Exit" << endl;**

**cout << "Enter the number: ";**

**cin >> n;**

**int price1 = 2000;**

**int price2 = 3000;**

**int price3 = 6000;**

**try**

**{**

**if(n==1)**

**{**

**if(Taken<=0)**

**{**

**cout<<"No more Games Left";**

**}**

**else{**

**Taken--;**

**GameShop::Amount(price1);**

**}**

**}**

**else if(n==2)**

**{**

**if(GtaV<=0)**

**{**

**cout<<"No more Games Left";**

**}**

**else{**

**GtaV--;**

**GameShop::Amount(price2);**

**}**

**}**

**else if(n==3)**

**{**

**if(Fifa23<=0)**

**{**

**cout<<"No more Games Left";**

**}**

**else{**

**Fifa23--;**

**GameShop::Amount(price3);**

**}**

**}**

**else if(n==4)**

**{**

**cout << "Thanks for visiting our shop..." << endl;**

**cout << "Please enter your phone number (length 11): ";**

**cin >> ph;**

**while (ph.length() > 11 || ph.length() < 10)**

**{**

**cout << "Phone number should consist of 10 or 11 digits." << endl;**

**cout << "Enter number: ";**

**cin >> ph;**

**}**

**}**

**else**

**{**

**throw(n);**

**}**

**}**

**catch(int num)**

**{**

**cout<<"Invalid choice."<< endl;**

**}**

**}**

**};**

**int GameShop:: Gow=2;**

**int GameShop:: Rdp=2;**

**int GameShop:: Fortnite=2;**

**class PS : public GameShop**

**{**

**public:**

**void p()**

**{**

**cout << "We have the following games on Play Station. Please choose one: " << endl;**

**cout << "1. God of War Stock: "<<Gow<<"\n2. Red Dead Redemption Stock:"<<Rdp<<"\n3. Fortnite Stock: "<<Fortnite<<"\n4. Exit" << endl;**

**cout << "Enter the number: ";**

**cin >> n;**

**int price1 = 5000;**

**int price2 = 7000;**

**int price3 = 9500;**

**try**

**{**

**if(n==1)**

**{**

**if(Gow<=0)**

**{**

**cout<<"No more Games Left";**

**}**

**else{**

**Gow--;**

**GameShop::Amount(price1);**

**}**

**}**

**else if(n==2)**

**{**

**if(Rdp<=0)**

**{**

**cout<<"No more Games Left";**

**}**

**else{**

**Rdp--;**

**GameShop::Amount(price2);**

**}**

**}**

**else if(n==3)**

**{**

**if(Fortnite<=0)**

**{**

**cout<<"No more Games Left";**

**}**

**else{**

**Fortnite--;**

**GameShop::Amount(price3);**

**}**

**}**

**else if(n==4)**

**{**

**cout << "Thanks for visiting our shop..." << endl;**

**cout << "Please enter your phone number (length 11): ";**

**cin >> ph;**

**while (ph.length() > 11 || ph.length() < 10)**

**{**

**cout << "Phone number should consist of 10 or 11 digits."<<endl;**

**cin>>ph;**

**}**

**}**

**else**

**{**

**throw(n);**

**}**

**}**

**catch(int num)**

**{**

**cout<<"Invalid choice."<< endl;**

**}**

**}**

**};**

**int main()**

**{**

**char opt;**

**GameShop obj1, obj2, total;**

**obj1.display();**

**do**

**{**

**obj2.get();**

**cout << "\n............................\n";**

**if (obj2.station\_type == "Xbox" || obj2.station\_type == "xbox")**

**{**

**xbox obj3;**

**obj3.x();**

**total = total + obj3; // Add the current purchase to the total**

**}**

**else if (obj2.station\_type == "PS" || obj2.station\_type == "ps")**

**{**

**PS obj4;**

**obj4.p();**

**total = total + obj4; // Add the current purchase to the total**

**}**

**else**

**{**

**cout << "Invalid Choice! Please Try Again." << endl;**

**}**

**cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";**

**cout << "\nDo you want to check other games? (Y/N)" << endl;**

**cin >> opt;**

**}**

**while (opt == 'Y' || opt == 'y');**

**cout << "Total Amount: Rs. " << total.pri << endl;**

**cout << "Thanks for visiting our shop..." << endl;**

**return 0;**

**}**

**Inheritance :**

The project employs a hierarchical structure using class inheritance, with a base class named "**Game shop**" and a derived class named "**Xbox** ."and the other derived class **“Ps4”** The "Game shop" class serves as the foundation for the console game shop , while the "**Xbox or Ps4**" class inherits the properties and methods from the "**Game shop**" class and adds specific functionalities related to car purchases and customer management.

**Program :**

The code is given below and is also available in the zip file with this report .

****

**Inheritance Relationships:**

**1. Base Class: Game shop**

Game shop class Contains general properties and methods related to the Game shop, including user data collection, console information display, and calculations for pricing and amounts and Defines static variables to keep track of car stock quantities.

**2. Derived Class: Xbox**

Car class Inherits from the "**Game shop**" class that extending its functionality adds an overloaded increment operator to keep track of the number of customers visiting the Game shop. It Implements a specific game selection mechanism, allowing customers to choose a game from the available stock and Manages customer data collection and purchase transactions, including verifying payment and updating stock quantities.

**3. Derived Class: Ps4**

Car class Inherits from the "**Game shop**" class that extending its functionality adds an overloaded increment operator to keep track of the number of customers visiting the Game shop. It Implements a specific game selection mechanism, allowing customers to choose a game from the available stock and Manages customer data collection and purchase transactions, including verifying payment and updating stock quantities.

**Usage and Functionality:**

**1.User Data Collection:**

The "**userdata**" method prompts the user to enter their personal information, such as phone number. Basic input validations are performed to ensure the correctness of phone numbers.

**3. Stock Management:**

The Static variables in the "**Game shop**" class keep track of the available stock quantities for each game. The stock quantities are decremented upon successful game purchases, ensuring accurate stock management.

**Operator Overlaoding:**

The code incorporates operator overloading to redefine the behavior of the increment (++) operator in relation to game objects. By overloading the increment operator, custom logic is implemented to increment the number of customers visiting the game shop, providing a more intuitive and expressive way to handle this operation.

**Overloading the Increment Operator:**

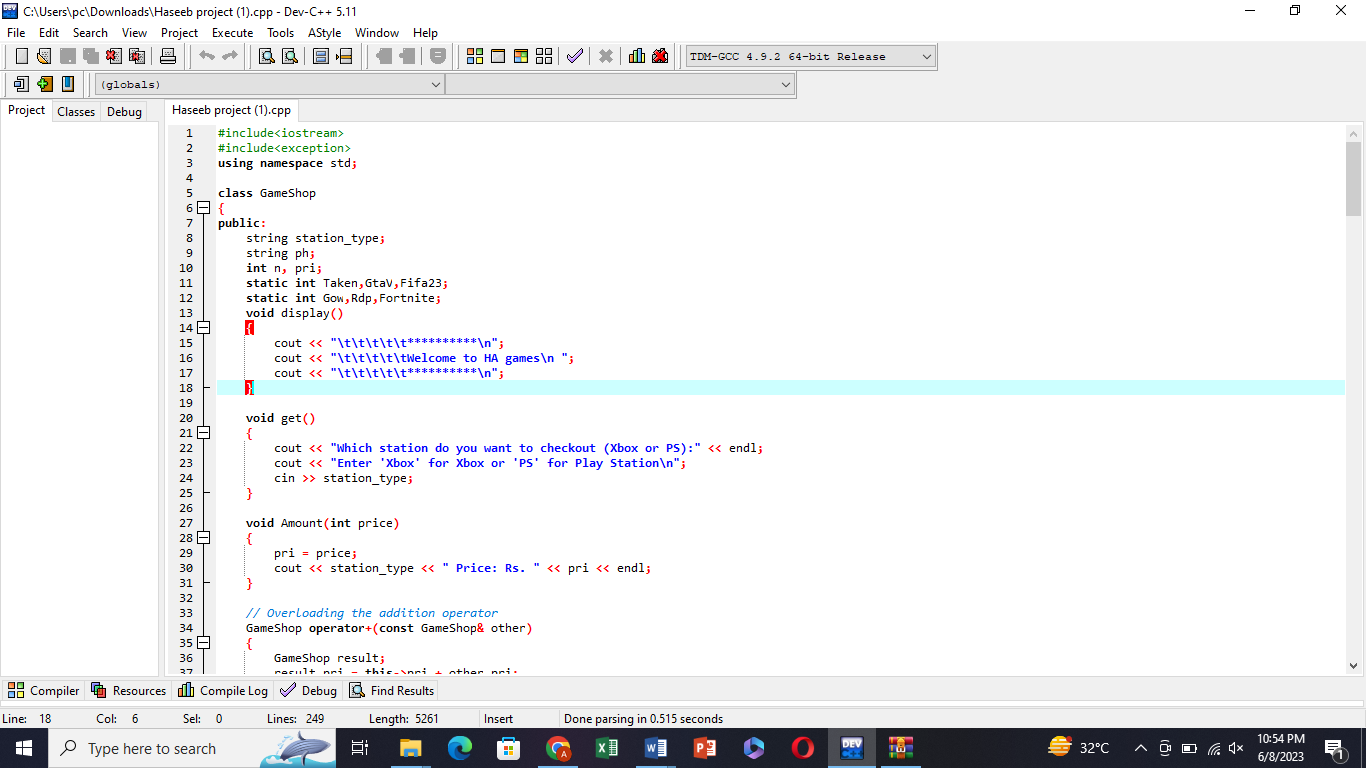
The "**xbox**" class overloads the increment (++) operator by defining the "**add**" method. This method is invoked when the increment operator is used on a xbox object. The "**add**" method increments the count of customers visiting the game shop, providing a concise and intuitive way to track the number of visitors.

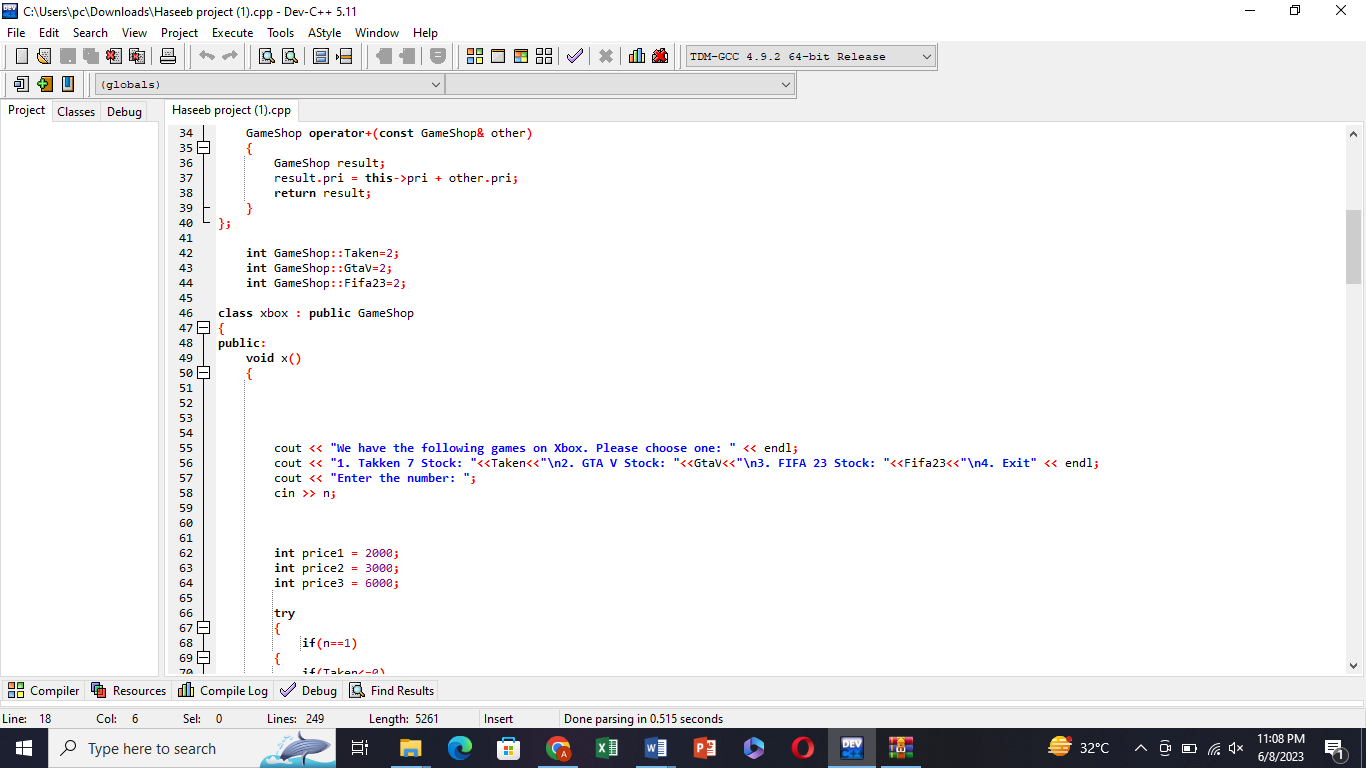
**Exceptional Handing:**

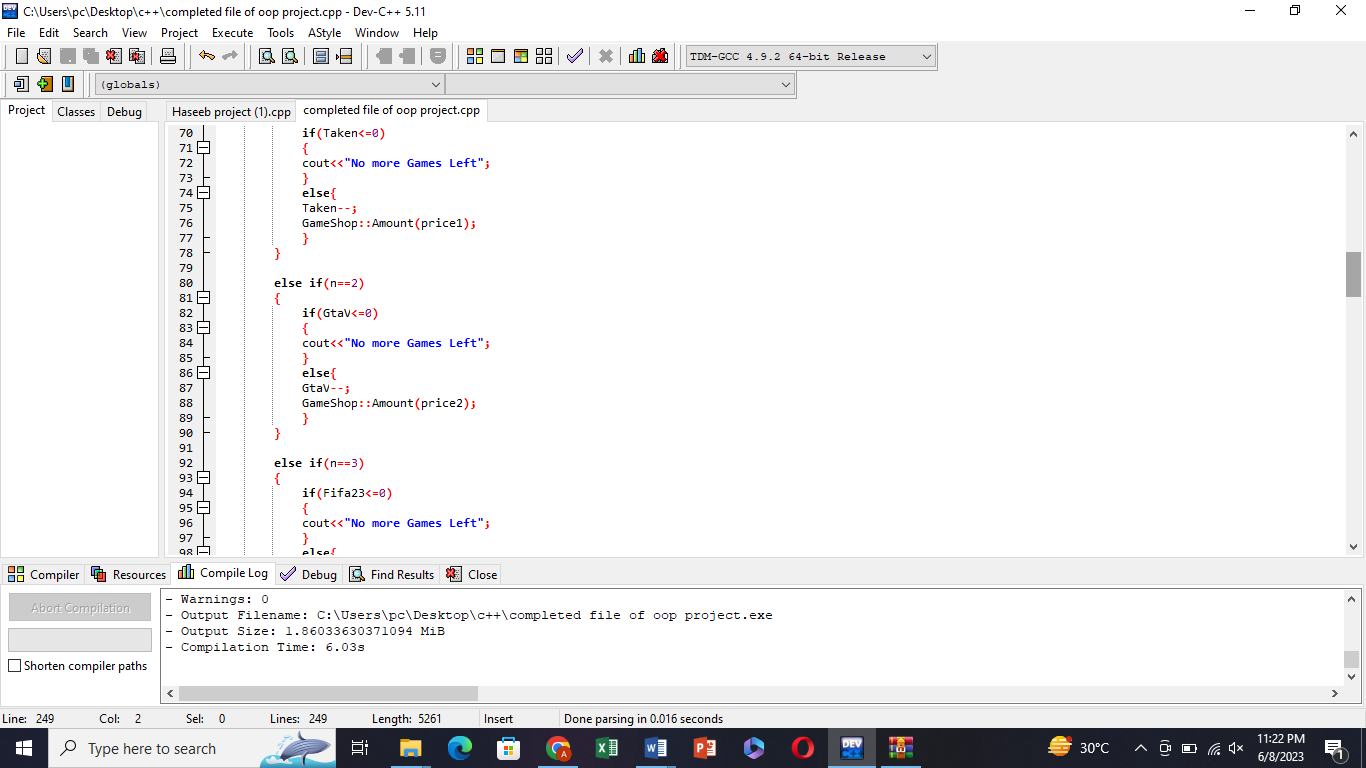
**Handling Invalid Input:**

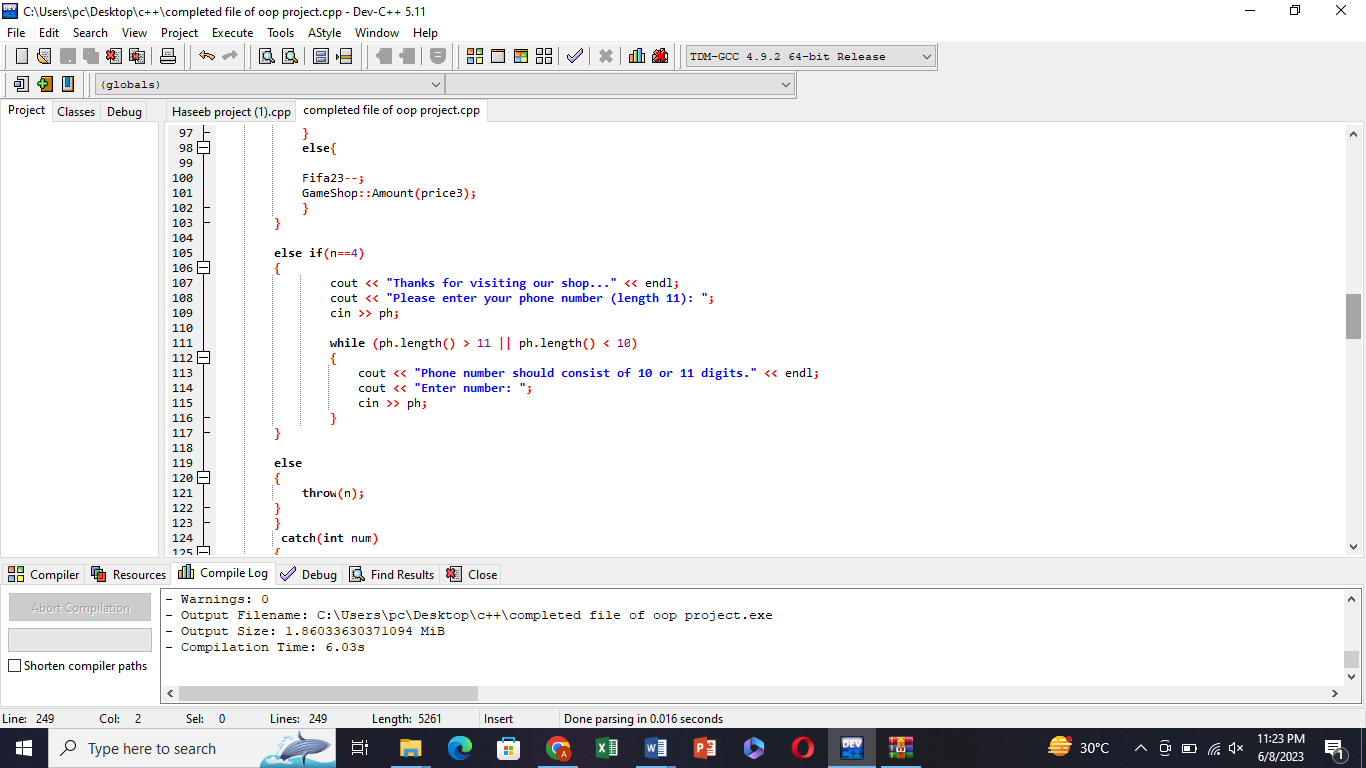
The code uses a try-except block to handle potential errors related to invalid input. Within the try block, user input for the car model is obtained and assigned to the "**Ps4**" variable. If the user enters an invalid game that is not present in the Game shop's game collection, a Value Error exception is raised. The except block catches the exception and provides appropriate feedback to the user.

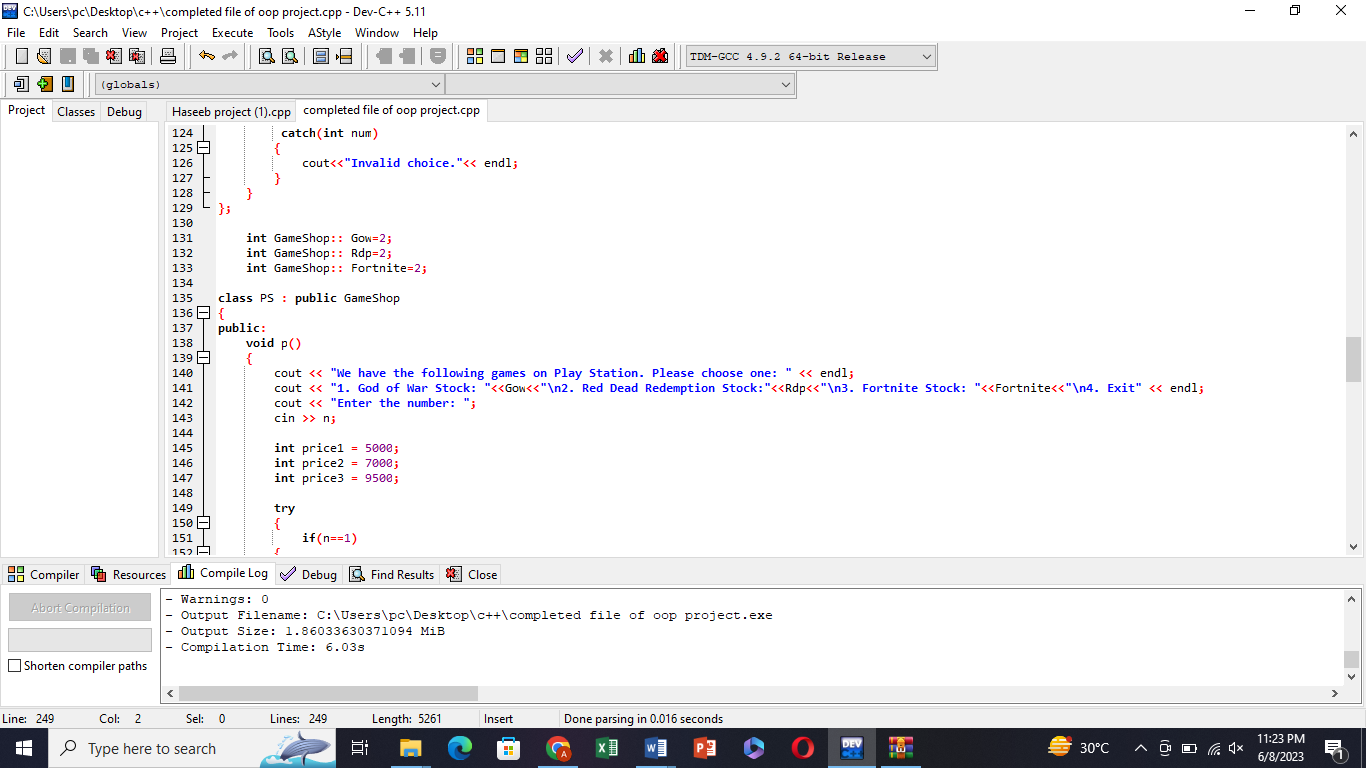
**Screenshots:**

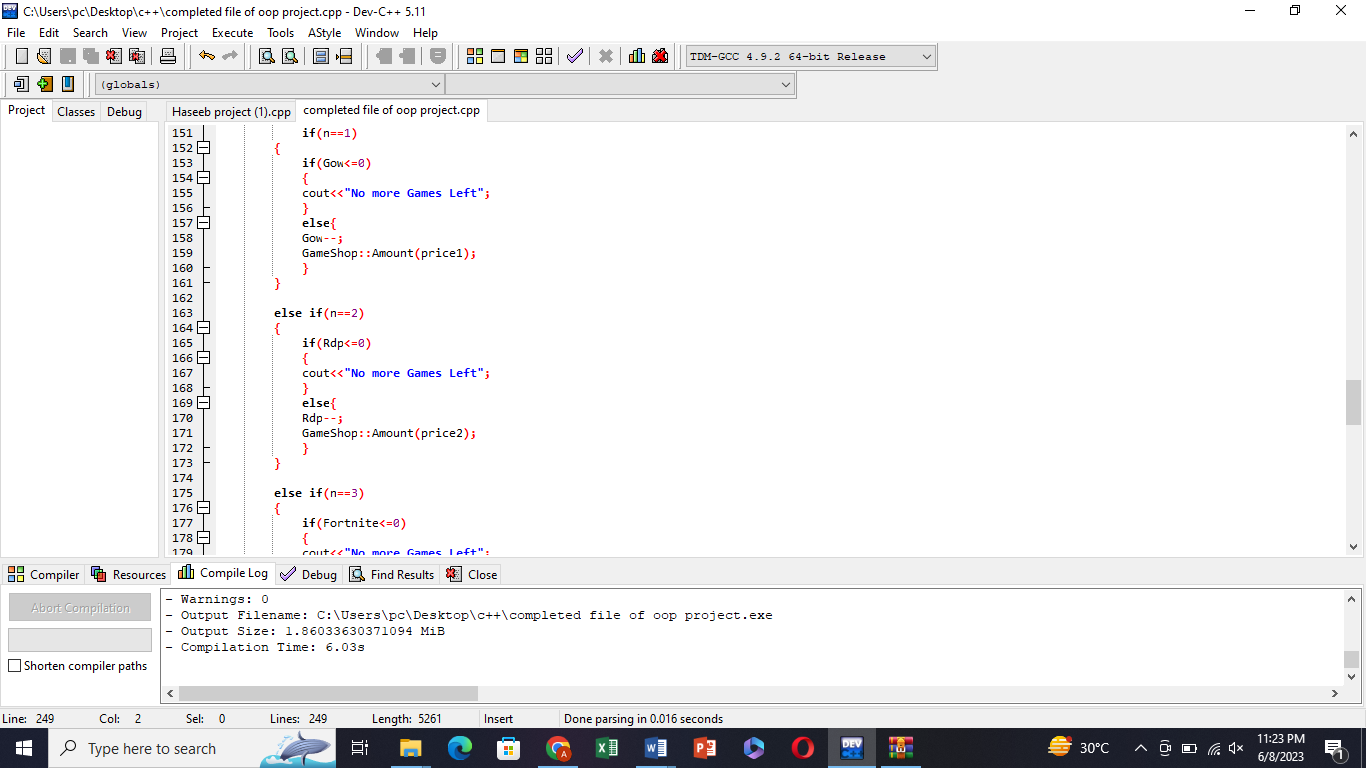


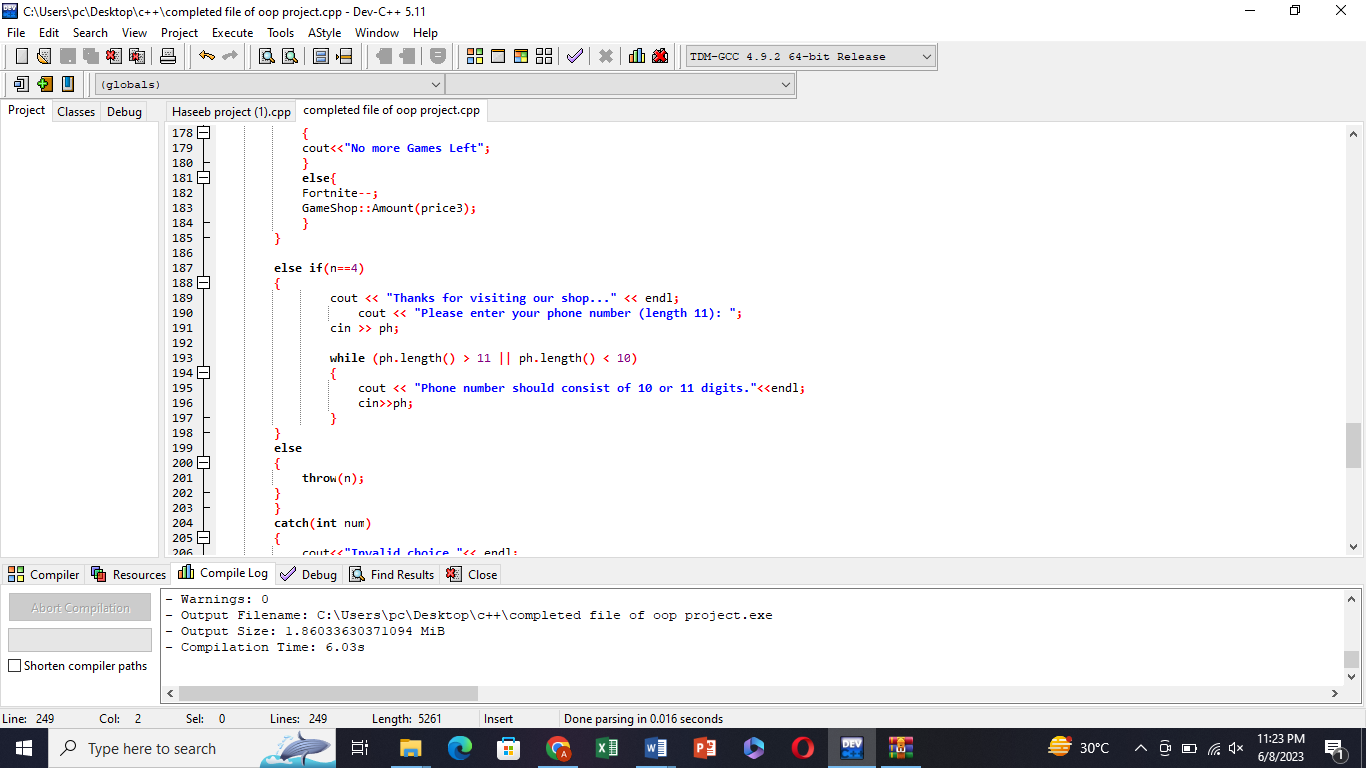


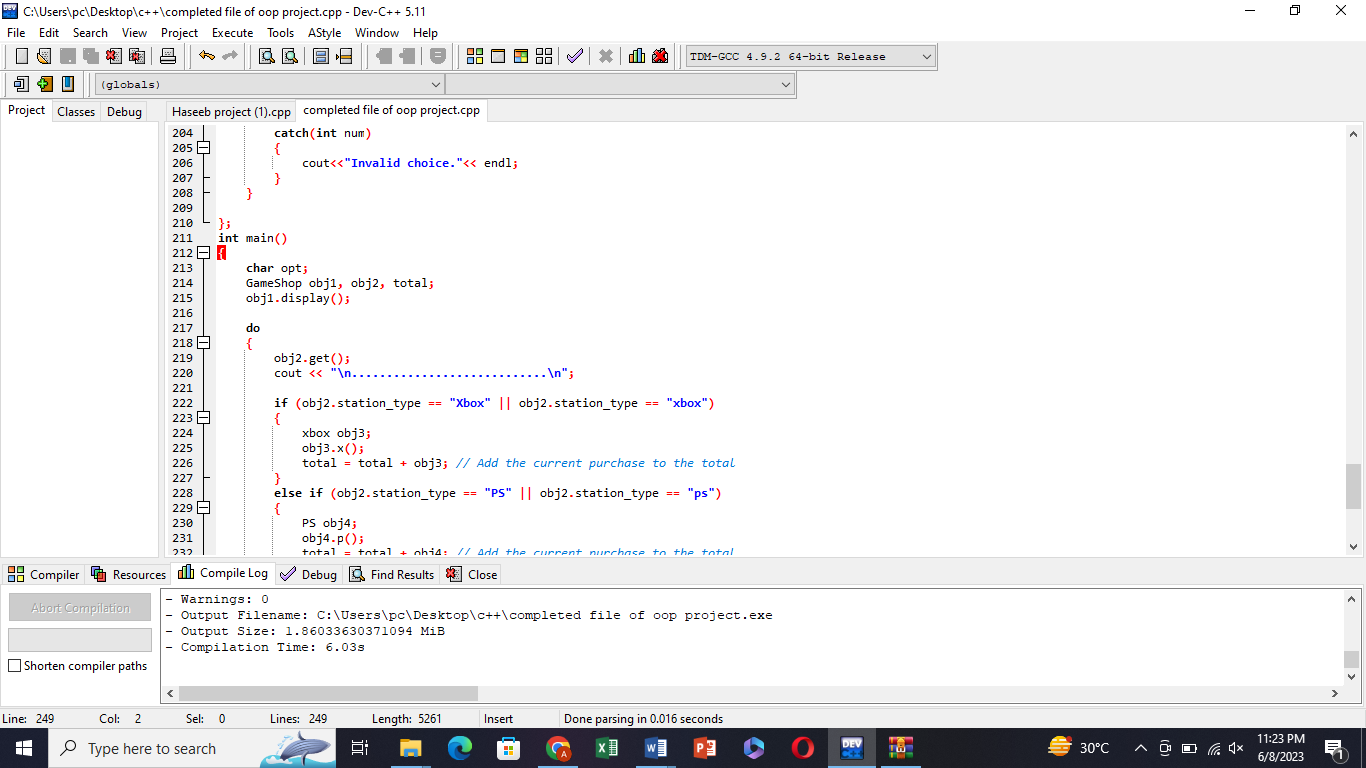


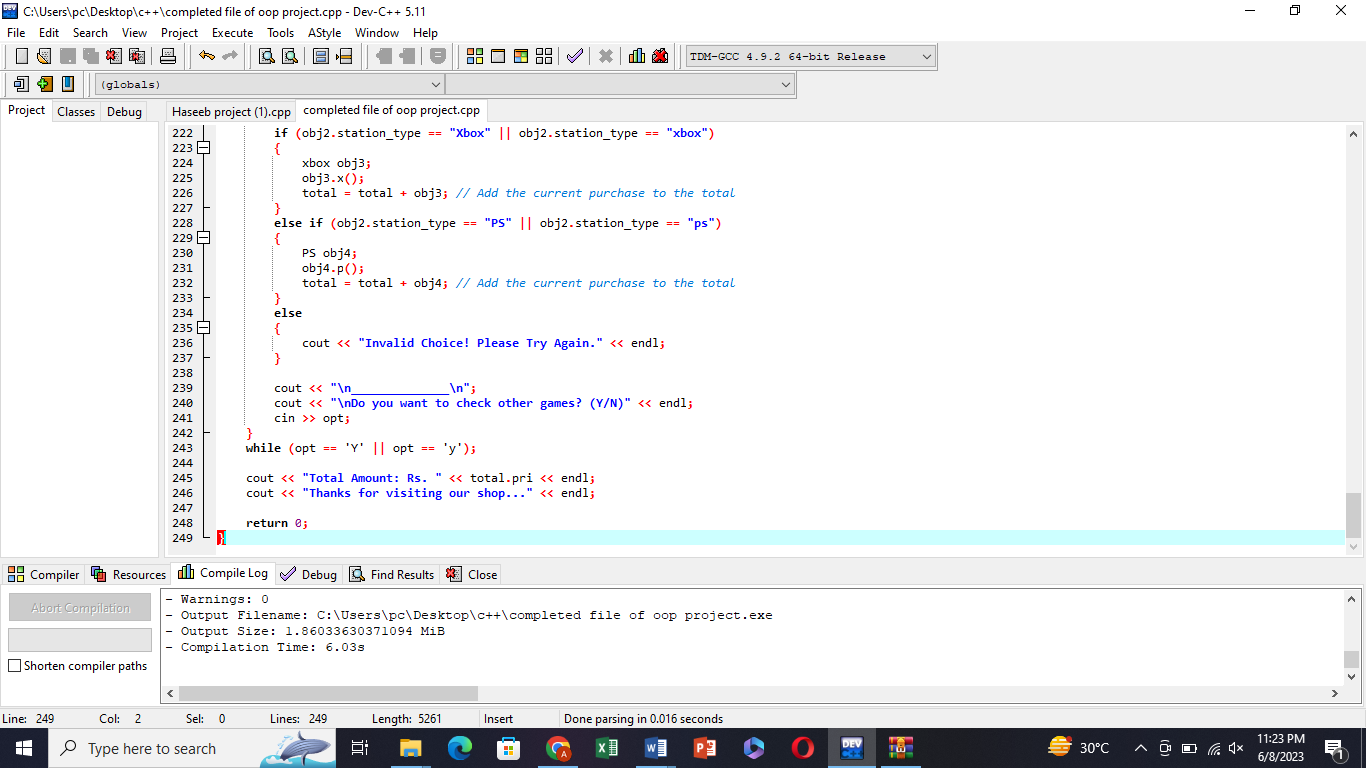




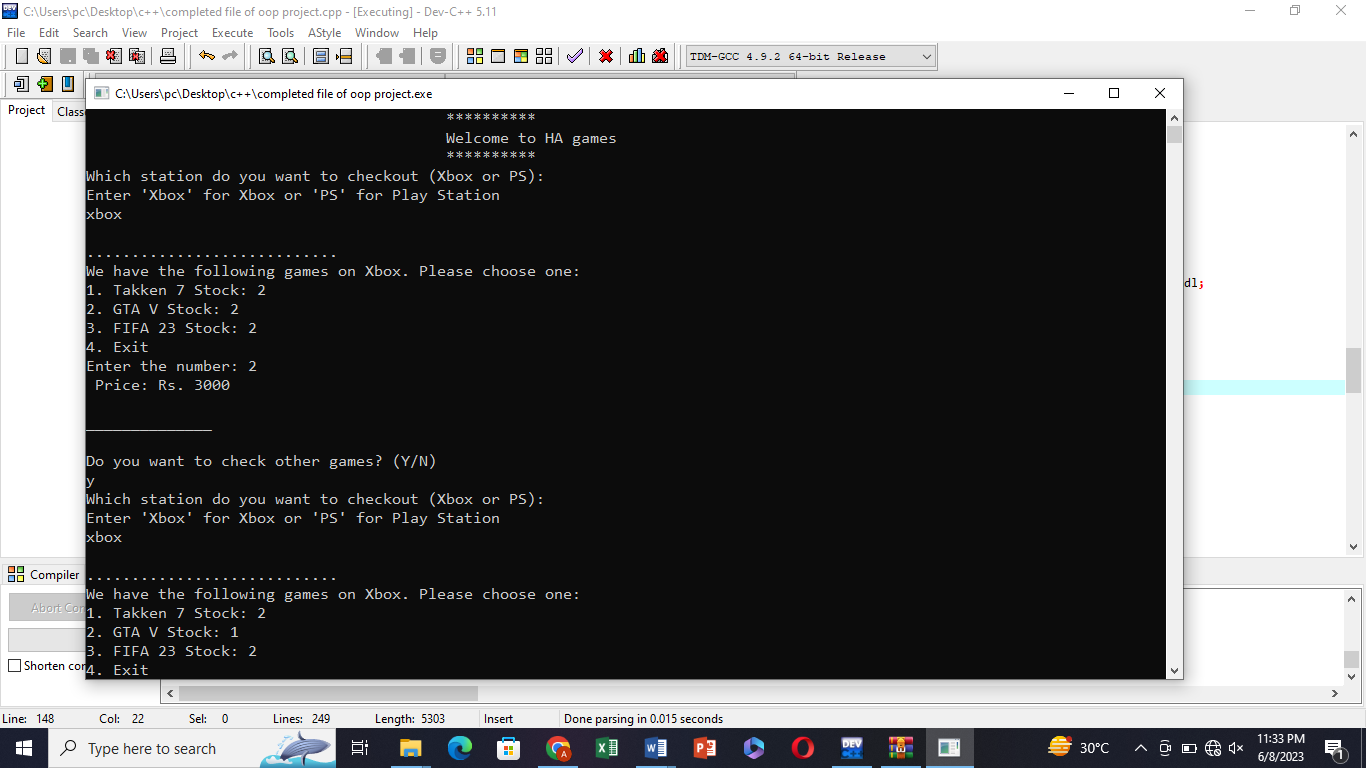


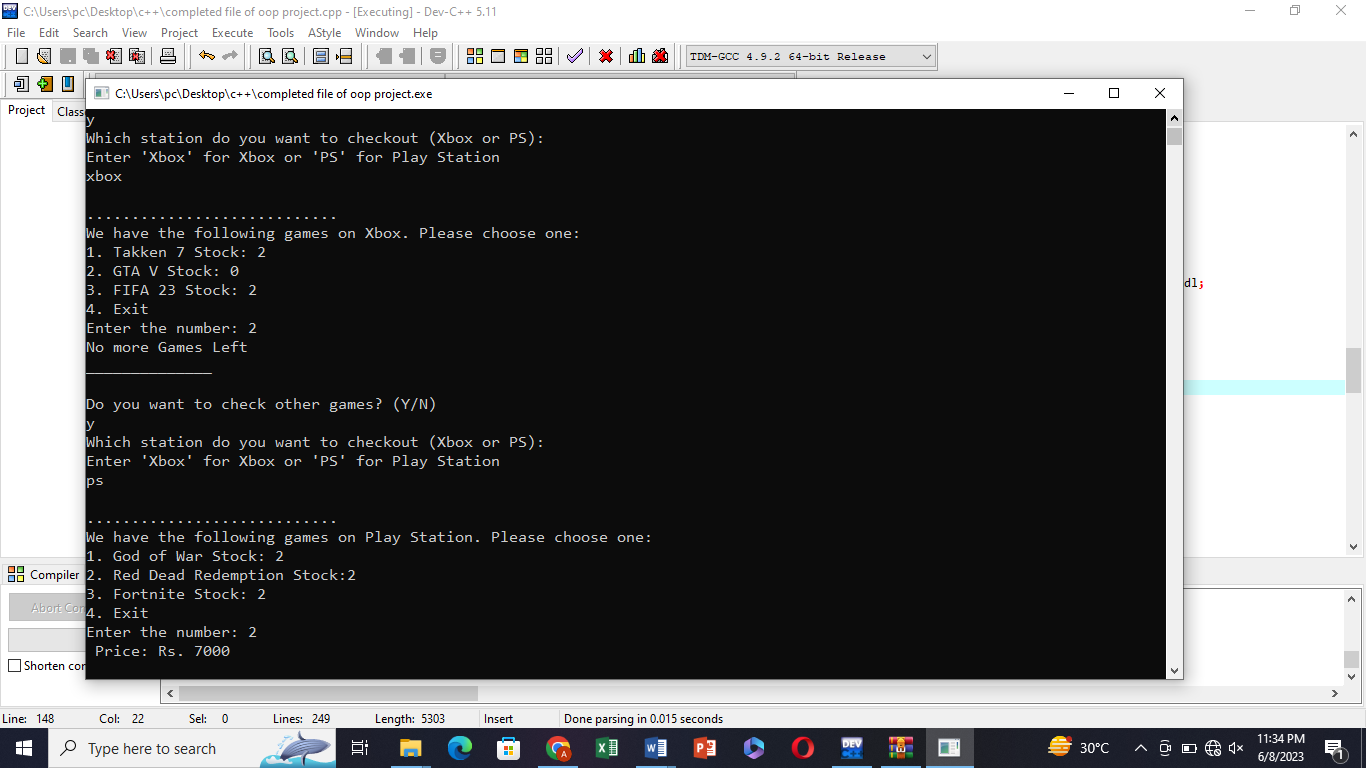


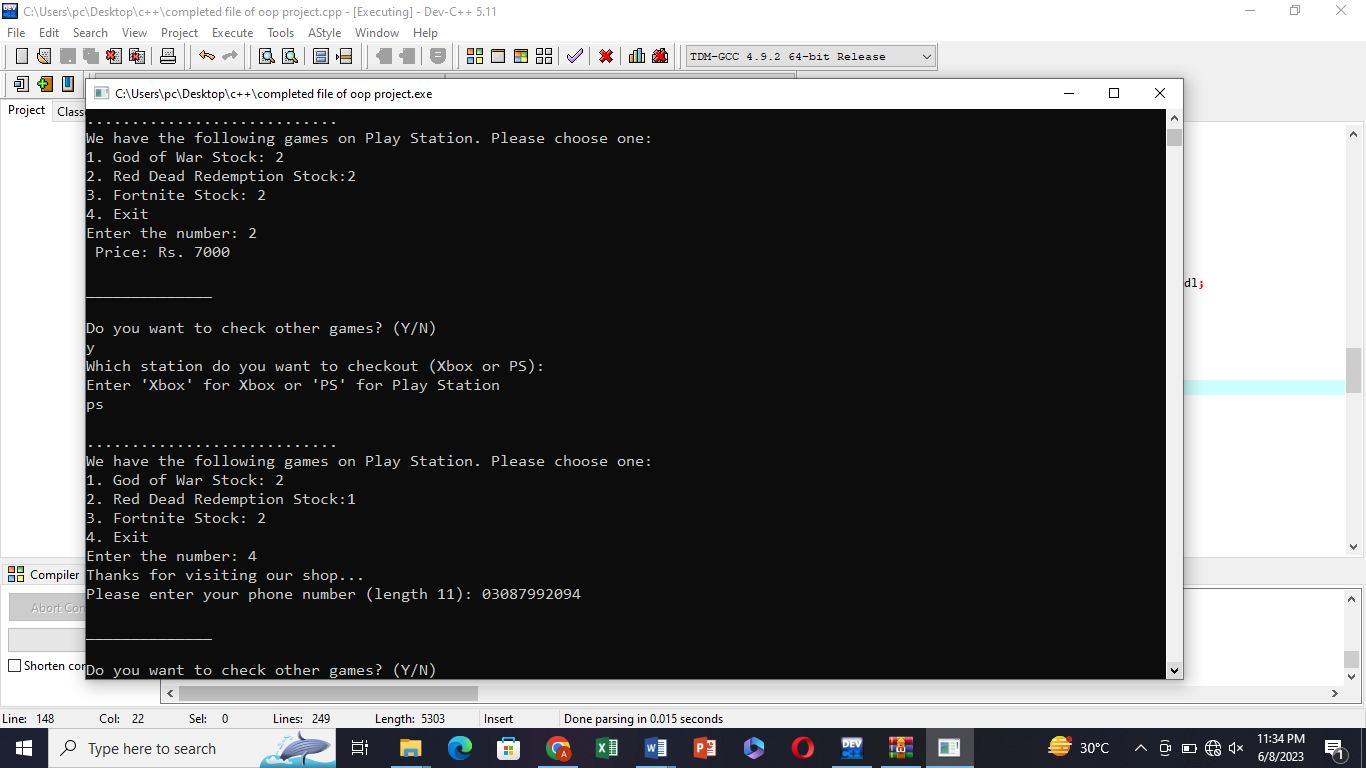


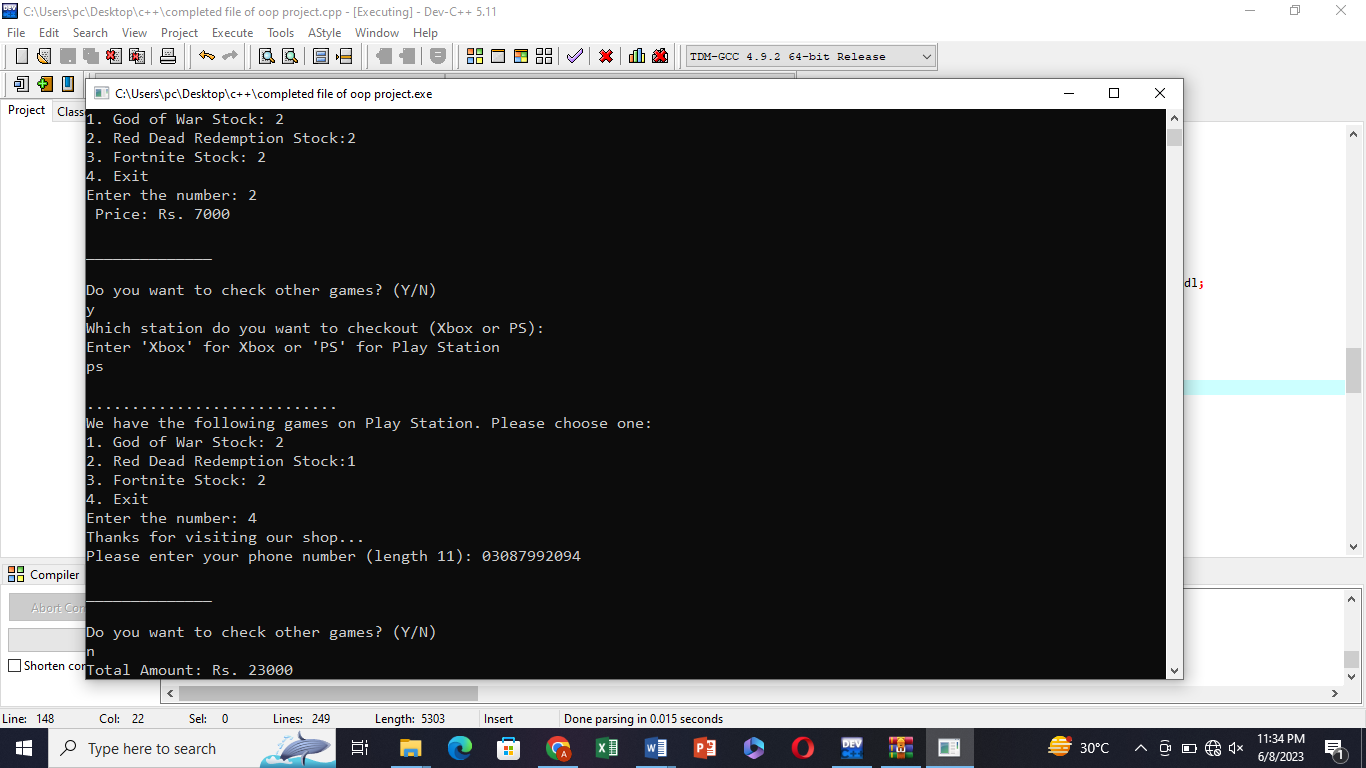


**Output :**

****

****

****

****

Thank u