Lab 5- Class (Advanced)

ELEC 3150 – Object Oriented Programming (Fall 2023)

Nick Cebula

Practice Question 1&2 (combined since using same class):

Q1 assumptions:

From Human class in slides 7-9, Create a class called “teacher”. This class is inherited from human class

- Attribute : Name, Bank_Account, Knowledge (max 100)
- Method:
  - Learn: Increase knowledge by 10, reduce money in bank by 20
  - Work: increase money in the bank by 50

In the main file (source.cpp), create a teacher object and show Money and knowledge of teacher when the following activities occur

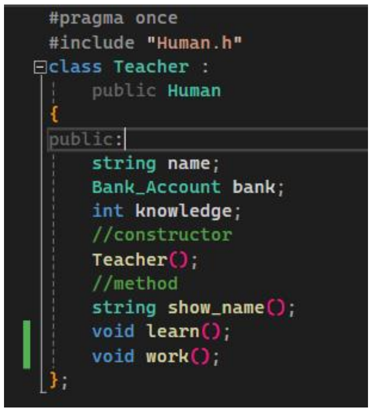
- Work >> Learn >> Learn >> Learn >> Work >> Learn

Q2 (inheritance) assumptions:

From previous question,

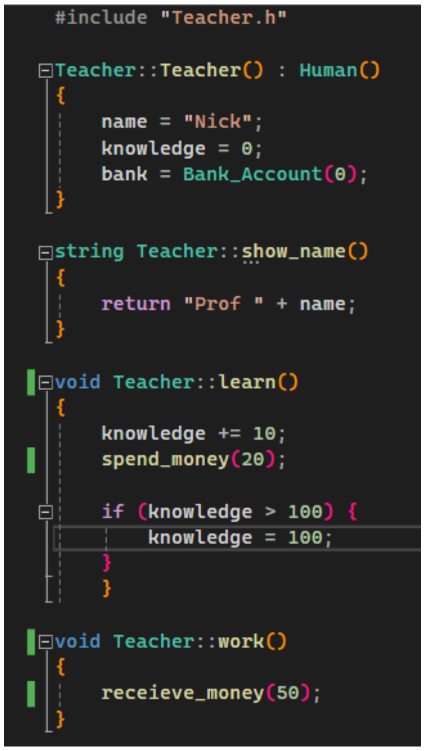
- Create a constructor for teacher class
- Rewrite showname method in teacher class, teacher’s name must start with Prof.
- Print name of teacher object on a screen

Teacher.h:



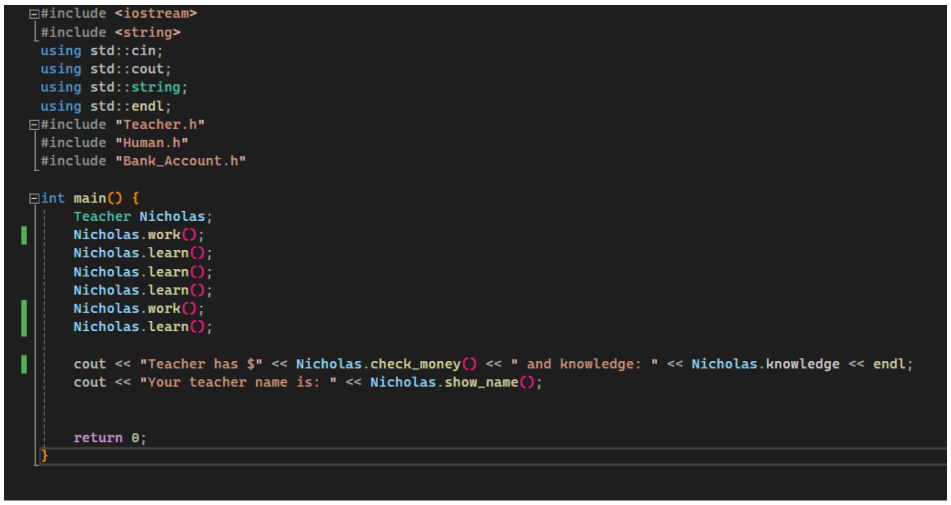
```
#pragma once
#include "Human.h"
class Teacher :
    public Human
{
public:
    string name;
    Bank_Account bank;
    int knowledge;
    //constructor
    Teacher();
    //method
    string show_name();
    void learn();
    void work();
};
```

Teacher.cpp:



```
#include "Teacher.h"

Teacher::Teacher() : Human()
{
    name = "Nick";
    knowledge = 0;
    bank = Bank_Account(0);
}

string Teacher::show_name()
{
    return "Prof " + name;
}

void Teacher::learn()
{
    knowledge += 10;
    spend_money(20);

    if (knowledge > 100) {
        knowledge = 100;
    }
    }

void Teacher::work()
{
    receieve_money(50);
}
```

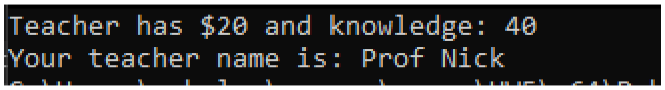
Main.cpp:



```
#include <iostream>
#include <string>
using std::cin;
using std::cout;
using std::string;
using std::endl;
#include "Teacher.h"
#include "Human.h"
#include "Bank_Account.h"

int main() {
    Teacher Nicholas;
    Nicholas.work();
    Nicholas.learn();
    Nicholas.learn();
    Nicholas.learn();
    Nicholas.work();
    Nicholas.learn();

    cout << "Teacher has $" << Nicholas.check_money() << " and knowledge: " << Nicholas.knowledge << endl;
    cout << "Your teacher name is: " << Nicholas.show_name();


    return 0;
}
```

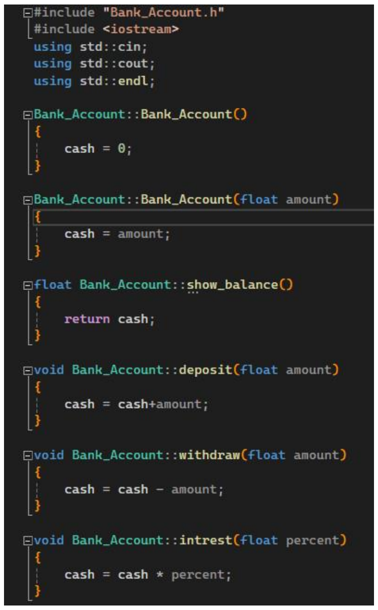
Description: The teacher header has attributes of name, uses inheritance from the Bank_Account to use its methods to transfer money, and then a int value for knowledge. I write a default constructor and method to show name, learn, and work. These methods in the cpp return name with a professor title, learn increases knowledge by 10, and uses method spendmoney to decrease money in bank by 20. Work method increases money in the bank by 50.

Results:



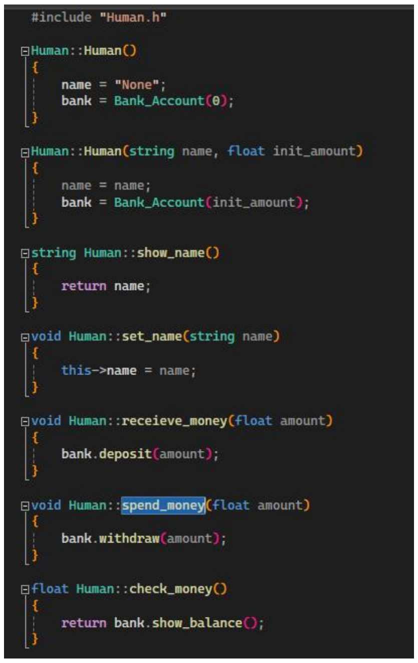
```
Teacher has $20 and knowledge: 40
Your teacher name is: Prof Nick
```

Bankaccount.cpp



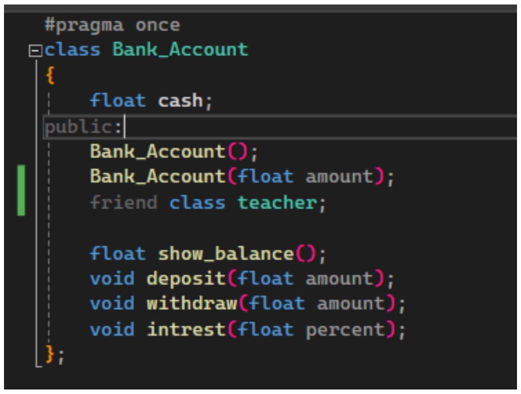
```cpp
#include "Bank_Account.h"
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

Bank_Account::Bank_Account()
{
    cash = 0;
}

Bank_Account::Bank_Account(float amount)
{
    cash = amount;
}

float Bank_Account::show_balance()
{
    return cash;
}

void Bank_Account::deposit(float amount)
{
    cash = cash+amount;
}

void Bank_Account::withdraw(float amount)
{
    cash = cash - amount;
}

void Bank_Account::intrest(float percent)
{
    cash = cash * percent;
}
```

Human.cpp:



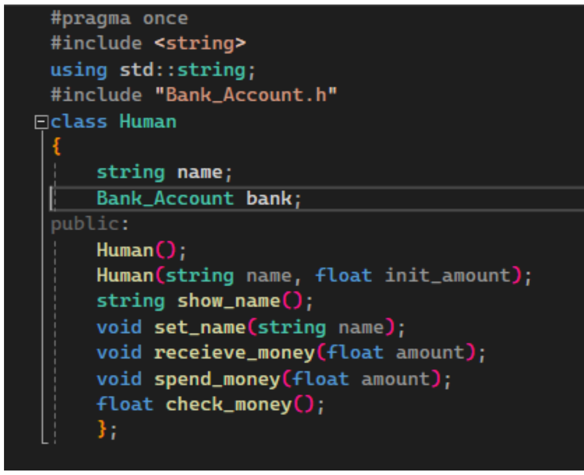
```cpp
#include "Human.h"

Human::Human()
{
    name = "None";
    bank = Bank_Account(0);
}

Human::Human(string name, float init_amount)
{
    name = name;
    bank = Bank_Account(init_amount);
}

string Human::show_name()
{
    return name;
}

void Human::set_name(string name)
{
    this->name = name;
}

void Human::receieve_money(float amount)
{
    bank.deposit(amount);
}

void Human::spend_money(float amount)
{
    bank.withdraw(amount);
}

float Human::check_money()
{
    return bank.show_balance();
}
```

Bankaccount.h:



```cpp
#pragma once
class Bank_Account
{
    float cash;
public:
    Bank_Account();
    Bank_Account(float amount);
    friend class teacher;

    float show_balance();
    void deposit(float amount);
    void withdraw(float amount);
    void intrest(float percent);
};
```

Human.h:

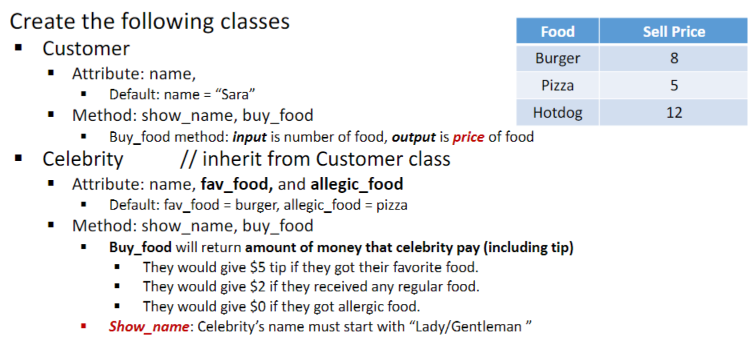


```cpp
#pragma once
#include <string>
using std::string;
#include "Bank_Account.h"
class Human
{
    string name;
    Bank_Account bank;
public:
    Human();
    Human(string name, float init_amount);
    string show_name();
    void set_name(string name);
    void receieve_money(float amount);
    void spend_money(float amount);
    float check_money();
};
```

Description: In the bank account it has attributes of a value cash, I made constructors for bank account and another constructor to receive amount of cash. It is friended with a class teacher so it's able to use its methods to transfer money. In the human class it has attributes of name and uses the class bank account it has methods of set name, receive money, spend money, check money. In the human class it has methods to show name set name receive money spend money and check money. The receive money spend money and check money use the bank account functions to deposit withdraw a certain amount of money in the check money function returns the the balance in the bank. At the end of the function after completing the commands in the main.cpp the teacher has $20 and knowledge of 40 this is after doing the series of activities in the assumptions it then prints the teachers name with the title of professor.

Lab question 1: Order Food

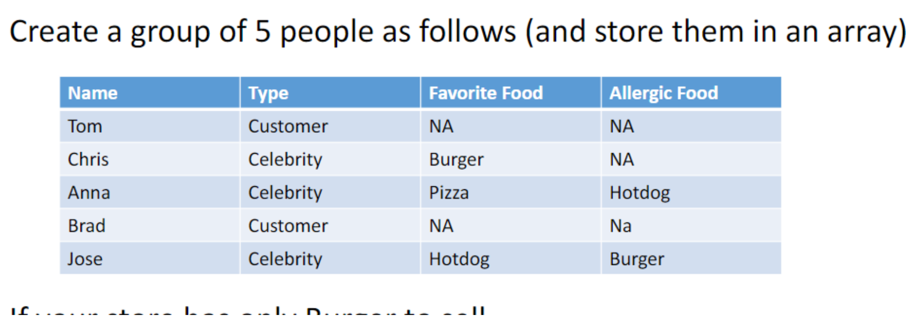
Assumptions:



Create the following classes

- Customer
  - Attribute: name,
    - Default: name = "Sara"
  - Method: show_name, buy_food
    - Buy_food method: ***input*** is number of food, ***output*** is ***price*** of food
- Celebrity // inherit from Customer class
  - Attribute: name, **fav_food,** and **allegic_food**
    - Default: fav_food = burger, allegic_food = pizza
  - Method: show_name, buy_food
    - **Buy_food** will return **amount of money that celebrity pay (including tip)**
      - They would give $5 tip if they got their favorite food.
      - They would give $2 if they received any regular food.
      - They would give $0 if they got allergic food.
    - ***Show_name***: Celebrity's name must start with "Lady/Gentleman "

Food	Sell Price
Burger	8
Pizza	5
Hotdog	12

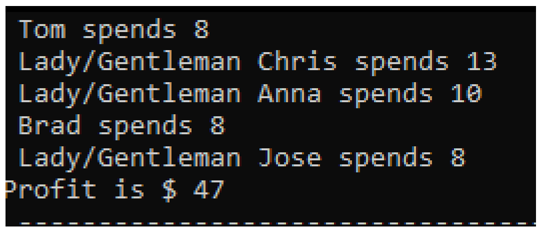


Create a group of 5 people as follows (and store them in an array)

Name	Type	Favorite Food	Allergic Food
Tom	Customer	NA	NA
Chris	Celebrity	Burger	NA
Anna	Celebrity	Pizza	Hotdog
Brad	Customer	NA	Na
Jose	Celebrity	Hotdog	Burger

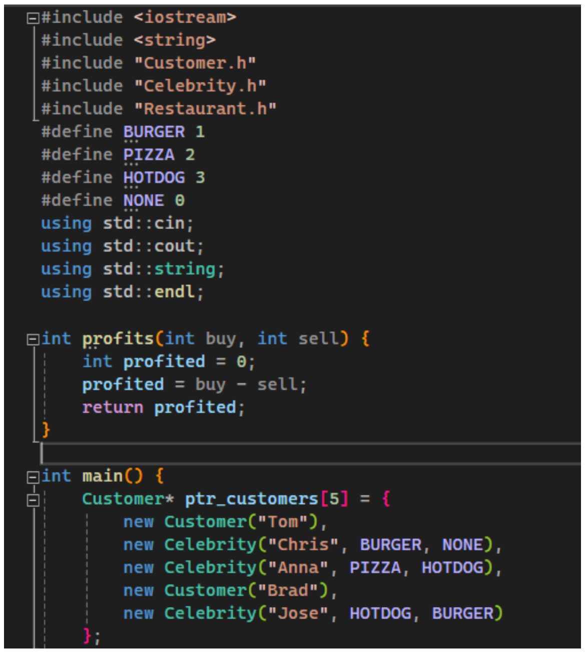
If your store has only Burger to sell

Results:

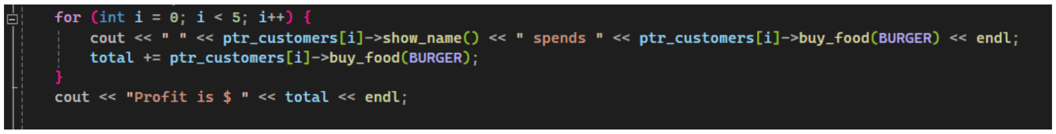


```
Tom spends 8
Lady/Gentleman Chris spends 13
Lady/Gentleman Anna spends 10
Brad spends 8
Lady/Gentleman Jose spends 8
Profit is $ 47
---------------------------------
```

Main.cpp:

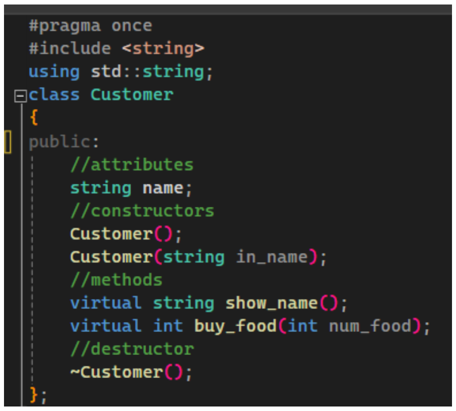


```cpp
#include <iostream>
#include <string>
#include "Customer.h"
#include "Celebrity.h"
#include "Restaurant.h"
#define BURGER 1
#define PIZZA 2
#define HOTDOG 3
#define NONE 0
using std::cin;
using std::cout;
using std::string;
using std::endl;

int profits(int buy, int sell) {
    int profited = 0;
    profited = buy - sell;
    return profited;
}

int main() {
    Customer* ptr_customers[5] = {
        new Customer("Tom"),
        new Celebrity("Chris", BURGER, NONE),
        new Celebrity("Anna", PIZZA, HOTDOG),
        new Customer("Brad"),
        new Celebrity("Jose", HOTDOG, BURGER)
    };
```



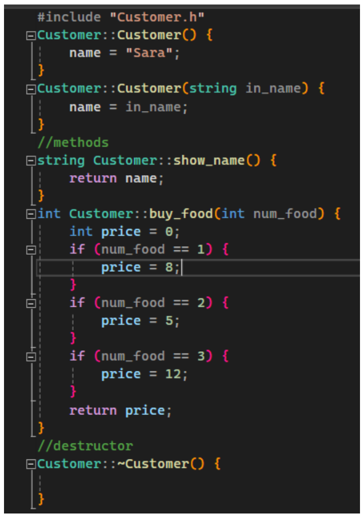
```cpp
    for (int i = 0; i < 5; i++) {
        cout << " " << ptr_customers[i]->show_name() << " spends " << ptr_customers[i]->buy_food(BURGER) << endl;
        total += ptr_customers[i]->buy_food(BURGER);
    }
    cout << "Profit is $ " << total << endl;
```

Customer.h:



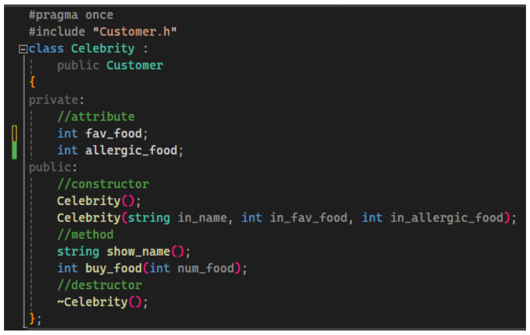
```cpp
#pragma once
#include <string>
using std::string;
class Customer
{
public:
    //attributes
    string name;
    //constructors
    Customer();
    Customer(string in_name);
    //methods
    virtual string show_name();
    virtual int buy_food(int num_food);
    //destructor
    ~Customer();
};
```

Customer.cpp:



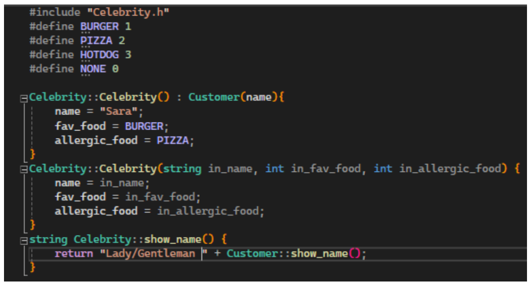
```cpp
#include "Customer.h"
Customer::Customer() {
    name = "Sara";
}
Customer::Customer(string in_name) {
    name = in_name;
}
//methods
string Customer::show_name() {
    return name;
}
int Customer::buy_food(int num_food) {
    int price = 0;
    if (num_food == 1) {
        price = 8;
    }
    if (num_food == 2) {
        price = 5;
    }
    if (num_food == 3) {
        price = 12;
    }
    return price;
}
//destructor
Customer::~Customer() {

}
```

Celebrity.h:



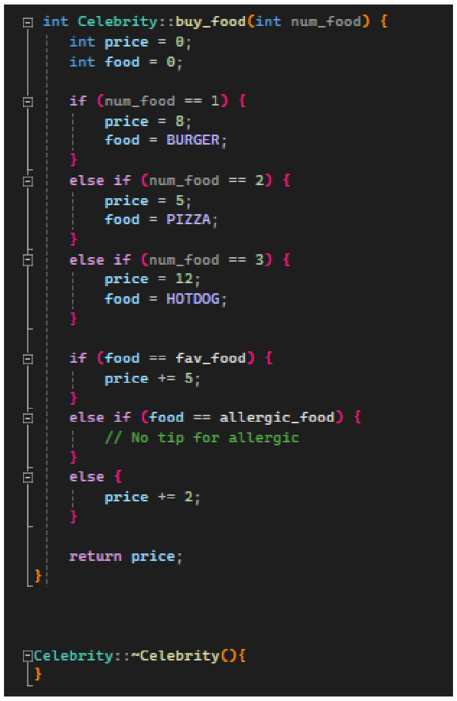
```cpp
#pragma once
#include "Customer.h"
class Celebrity :
    public Customer
{
private:
    //attribute
    int fav_food;
    int allergic_food;
public:
    //constructor
    Celebrity();
    Celebrity(string in_name, int in_fav_food, int in_allergic_food);
    //method
    string show_name();
    int buy_food(int num_food);
    //destructor
    ~Celebrity();
};
```

Celebrity.cpp



```cpp
#include "Celebrity.h"
#define BURGER 1
#define PIZZA 2
#define HOTDOG 3
#define NONE 0

Celebrity::Celebrity() : Customer(name){
    name = "Sara";
    fav_food = BURGER;
    allergic_food = PIZZA;
}
Celebrity::Celebrity(string in_name, int in_fav_food, int in_allergic_food) {
    name = in_name;
    fav_food = in_fav_food;
    allergic_food = in_allergic_food;
}
string Celebrity::show_name() {
    return "Lady/Gentleman " + Customer::show_name();
}
```

Celebrity.cpp continued:



```
int Celebrity::buy_food(int num_food) {
    int price = 0;
    int food = 0;

    if (num_food == 1) {
        price = 8;
        food = BURGER;
    }
    else if (num_food == 2) {
        price = 5;
        food = PIZZA;
    }
    else if (num_food == 3) {
        price = 12;
        food = HOTDOG;
    }

    if (food == fav_food) {
        price += 5;
    }
    else if (food == allergic_food) {
        // No tip for allergic
    }
    else {
        price += 2;
    }

    return price;
}


Celebrity::~Celebrity(){
}
```

Description: in the main dot CPP I created an array on the stack to store all of the customers in their foods I then made a for loop to cover all the customers and what they spend the customer method we'll buy food and based on which food they buy it will simply set the price of the food to what it costs and return the price. Whereas the celebrity class does that but then returns the price plus tip if they receive their favorite food. If they get an allergic food there's no tip otherwise they just receive a $2 tip in the main when you print the names it will say if they are celebrity or not based on the title for celebrity which is lady/ gentlemen which is done by inheritance.

Question 2: Dining Frenzy

Assumptions:

In this game, user is an owner of café. This cafe sells Burger, Pizza, and Hotdog.

- At the beginning, user have $50

Every morning, user must order food to stock their product.

- User can order food only once a day
- Once a café is open, user is not allowed to order any more food.

User must serve one food for each customer

- If user doesn't serve any food to customer, café must pay $10 fine.
- Each customer has different preference. If user serves customer's favorite food, they will get tips (Tip can be found from customer and celebrity class in question 1)

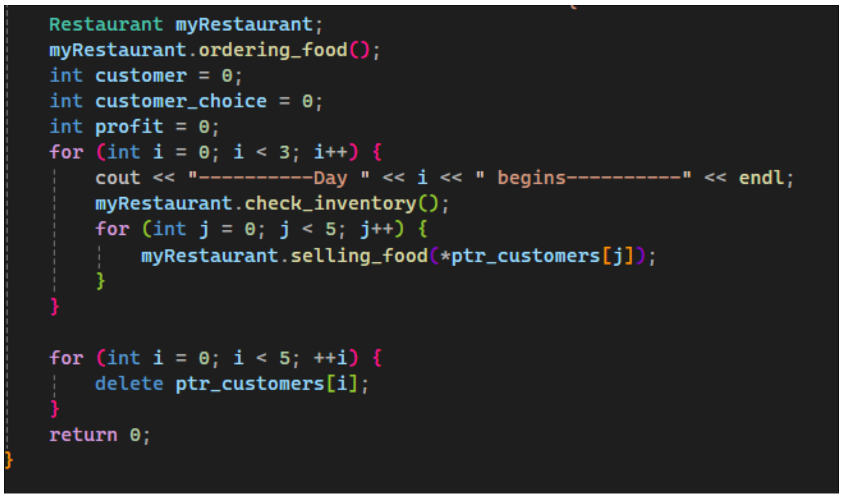
Assuming the same group of customers (in question 1) come to this café everyday. Play this game for 3 days and find total money that this café make.

Create a class called Restaurant

Food	Cost ($)	Sell Price
Burger	5	8
Pizza	3	5
Hotdog	7	12

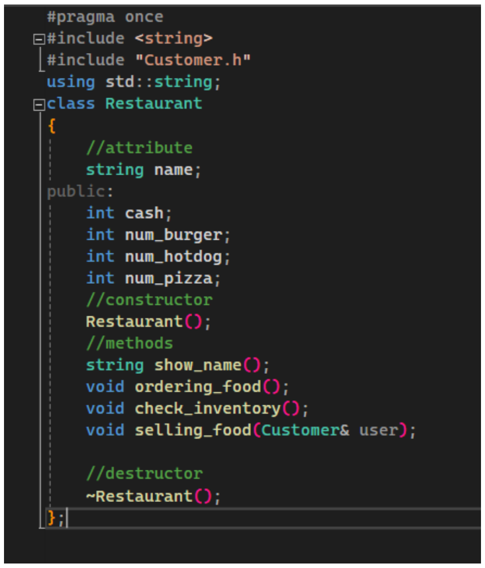
- Attribute:
  - Name, cash, num_burger, num_hotdog, num_pizza
- Constructor:
  - Default: cash = $50, num_burger = 0, num_hotdog = 0, num_pizza = 0
- Method
  - Show_name: // show name of restaurant follow by ***Inc***.
  - Ordering_food // no input, no output
    - Order burger, pizza, and hotdog for selling at a restaurant
    - This method can be called only once a day. You can't spend money more than you have. The cost of each food is shown in a table above
  - Check inventory
    - Print number of burger, pizza, and hotdog in your inventory
  - Selling_food // input is a customer, output is none
    - Sell food to customer. Customer will pay price of food plus tip to a restaurant
    - If customer didn't get food, restaurant must pay $10 fine for a city.

Main.cpp:



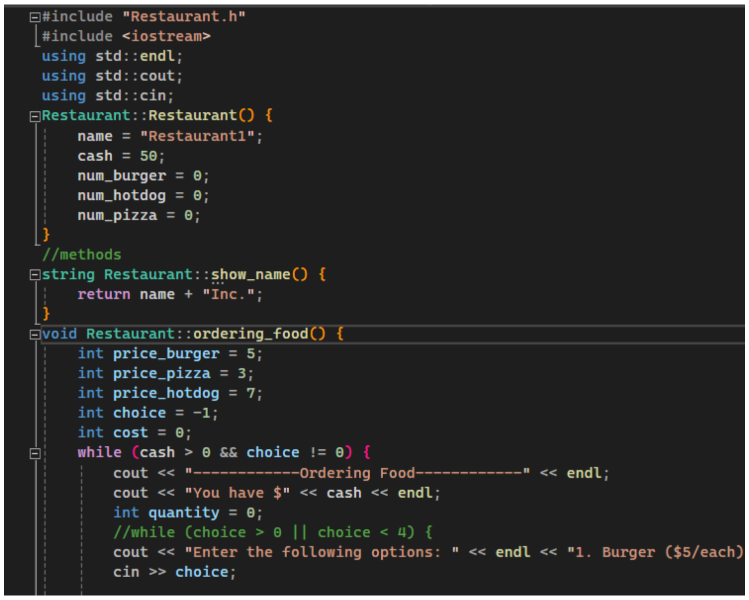
```
    Restaurant myRestaurant;
    myRestaurant.ordering_food();
    int customer = 0;
    int customer_choice = 0;
    int profit = 0;
    for (int i = 0; i < 3; i++) {
        cout << "----------Day " << i << " begins----------" << endl;
        myRestaurant.check_inventory();
        for (int j = 0; j < 5; j++) {
            myRestaurant.selling_food(*ptr_customers[j]);
        }
    }

    for (int i = 0; i < 5; ++i) {
        delete ptr_customers[i];
    }
    return 0;
}
```

Resturant.h:



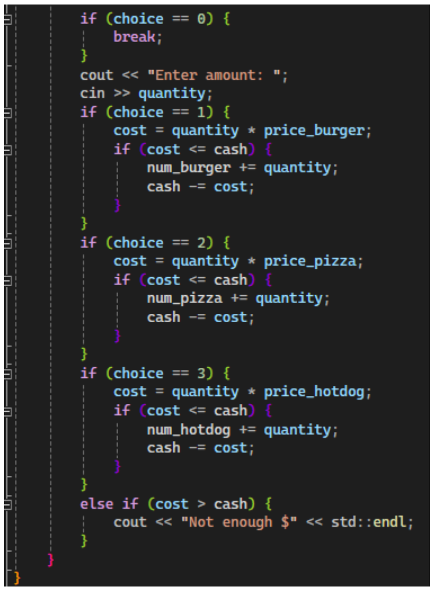
```
#pragma once
#include <string>
#include "Customer.h"
using std::string;
class Restaurant
{
    //attribute
    string name;
public:
    int cash;
    int num_burger;
    int num_hotdog;
    int num_pizza;
    //constructor
    Restaurant();
    //methods
    string show_name();
    void ordering_food();
    void check_inventory();
    void selling_food(Customer& user);

    //destructor
    ~Restaurant();
};
```

Resturant.cpp:

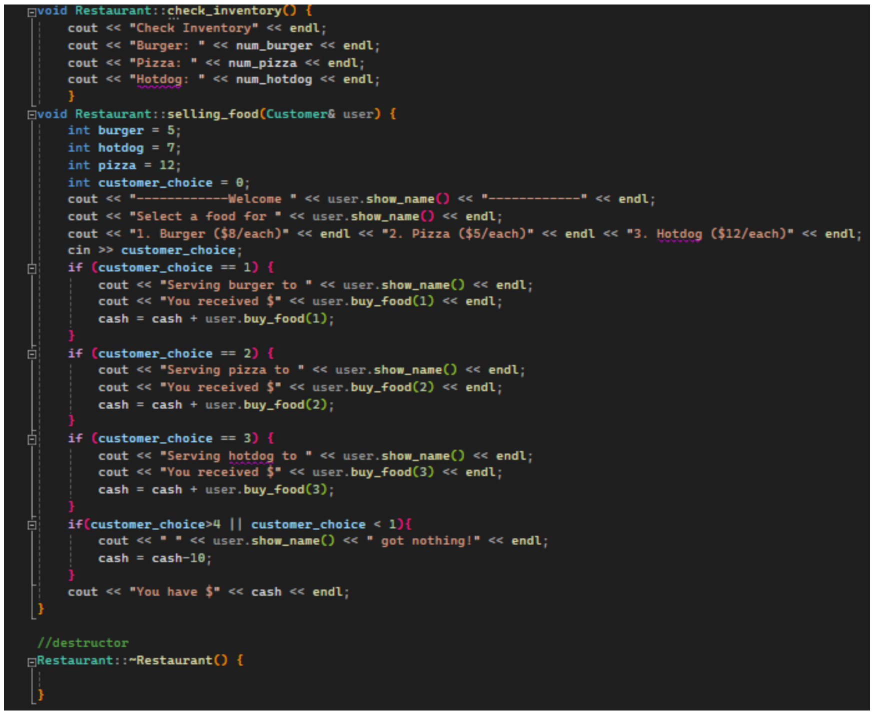


```
#include "Restaurant.h"
#include <iostream>
using std::endl;
using std::cout;
using std::cin;
Restaurant::Restaurant() {
    name = "Restaurant1";
    cash = 50;
    num_burger = 0;
    num_hotdog = 0;
    num_pizza = 0;
}
//methods
string Restaurant::show_name() {
    return name + "Inc.";
}
void Restaurant::ordering_food() {
    int price_burger = 5;
    int price_pizza = 3;
    int price_hotdog = 7;
    int choice = -1;
    int cost = 0;
    while (cash > 0 && choice != 0) {
        cout << "------------Ordering Food------------" << endl;
        cout << "You have $" << cash << endl;
        int quantity = 0;
        //while (choice > 0 || choice < 4) {
        cout << "Enter the following options: " << endl << "1. Burger ($5/each)
        cin >> choice;
```

Restaurant.cpp continued:



```cpp
        if (choice == 0) {
            break;
        }
        cout << "Enter amount: ";
        cin >> quantity;
        if (choice == 1) {
            cost = quantity * price_burger;
            if (cost <= cash) {
                num_burger += quantity;
                cash -= cost;
            }
        }
        if (choice == 2) {
            cost = quantity * price_pizza;
            if (cost <= cash) {
                num_pizza += quantity;
                cash -= cost;
            }
        }
        if (choice == 3) {
            cost = quantity * price_hotdog;
            if (cost <= cash) {
                num_hotdog += quantity;
                cash -= cost;
            }
        }
        else if (cost > cash) {
            cout << "Not enough $" << std::endl;
        }
    }
}
```



```cpp
void Restaurant::check_inventory() {
    cout << "Check Inventory" << endl;
    cout << "Burger: " << num_burger << endl;
    cout << "Pizza: " << num_pizza << endl;
    cout << "Hotdog: " << num_hotdog << endl;
    }
void Restaurant::selling_food(Customer& user) {
    int burger = 5;
    int hotdog = 7;
    int pizza = 12;
    int customer_choice = 0;
    cout << "------------Welcome " << user.show_name() << "------------" << endl;
    cout << "Select a food for " << user.show_name() << endl;
    cout << "1. Burger ($8/each)" << endl << "2. Pizza ($5/each)" << endl << "3. Hotdog ($12/each)" << endl;
    cin >> customer_choice;
    if (customer_choice == 1) {
        cout << "Serving burger to " << user.show_name() << endl;
        cout << "You received $" << user.buy_food(1) << endl;
        cash = cash + user.buy_food(1);
    }
    if (customer_choice == 2) {
        cout << "Serving pizza to " << user.show_name() << endl;
        cout << "You received $" << user.buy_food(2) << endl;
        cash = cash + user.buy_food(2);
    }
    if (customer_choice == 3) {
        cout << "Serving hotdog to " << user.show_name() << endl;
        cout << "You received $" << user.buy_food(3) << endl;
        cash = cash + user.buy_food(3);
    }
    if(customer_choice>4 || customer_choice < 1){
        cout << " " << user.show_name() << " got nothing!" << endl;
        cash = cash-10;
    }
    cout << "You have $" << cash << endl;
}

//destructor
Restaurant::~Restaurant() {

}
```

Description: For part 2 it was pretty simple, I made a class called restaurant which kept track of number of foods as an attribute and has cash. The methods are used to check inventory and print the current inventory. The selling food menu prints based on customer choice and serves the customer and prints the amount of money received keep in mind this uses inheritance of celebrity favorite food. The ordering food menu lets the user order food before the day begins based on the restaurant balance. You are able to stock the inventory.

Self Assessment: This lab was really fun making the dining game, serving the customers in order making the methods based on inheritance was very useful, you can put the whole cout statements and it really simplifies the main.cpp code. It allows the method to use the attributes based on the case and manage cash and inventory amounts.