HW Submission requirements:
1) Put your name and ID number on the top of EACH assignment (in comments for programs)
2) Name each file the name written in blue above each problem
3) Put all the files into a folder and zip it
3) Name the zipped folder with all assignments HW#.zip (-5 points for incorrect name)

NOTES:
1. **INCLUDE A README FOR EACH PROGRAM *(-10 points) Remember a README should let the user (my TA) know how to run your program***
2. **MAKE SURE YOUR PROGRAM IS PROPERLY INDENTED. *(-10 points)***
3. **MAKE SURE ANY COMMENTS YOU INCLUDE ARE MEANINGFUL *(-10 points) Do not just put random comments on every line of code***
4. **DO NOT CODE EVERYTHING IN MAIN *(Automatic 0)***

-INCLUDE UML DIAGRAMS (ACTIVITY AND CLASS) FOR EACH PROGRAM.

Problem 1 (20 points)-True/False Submit a document called Answers.doc

*Answer the following true/false questions. You must correctly state **WHY** your answer is true or false in order to receive credit.*

```
#include <iostream>
using namespace std;

class Food{

 protected:
  float price;

 public:
  void give_info(float price)
  {
     this->price=price;
  }

  float get_price(){
   return price;
  }

  void change_price(float a)
  {
   price=a;
  }
```

```
};

class Watermelon: public Food{
  bool seedless;

  public:

  void set_seed(int a)
  {
    if(a<3)
    {
      seedless=true;
    }
    else
    {
      seedless=false;
    }
  }

  bool get_seedless()
  {
    return seedless;
  }

};

class Apple: protected Food{


};

class Grape: private Food{

  string color;
public:
  int number;

};

class Employee{

public:
  void discount(Food &f1)
  {
     float amount;
```

```
    cout <<"How much of a discount? (Enter in decimal form)"<<endl;
    cin>>amount;

    f1.change_price(f1.get_price()*(1-amount));
  }

  //true=both, false=not both
  bool check_watermelon(Watermelon w1, Watermelon w2)
  {
   bool ret=false;

   if(w1.get_seedless() && w2.get_seedless())
   {
      ret=true;
   }
   return ret;
  }

};
```

For the following questions, assume any lines of code given are in the main function. Also assume: ***w1*** *refers to a Watermelon object,* ***a1*** *refers to an Apple object,* ***f1*** *refers to a Food object,* ***g1*** *refers to a Grape object and* ***e1*** *refers to an Employee object.*

1. *Watermelon w1;* would be a valid line of code, but *Apple a1;* would not be.
2. *e1.discount(w1);* would be a valid line of code.
3. *e1.discount(f1);* would be a valid line of code.
4. The function *set_seed()* in the Watermelon class is unnecessary since we could just say *w1.seedless=true;* .
5. *a1.give_info(2.99);* would be a valid line of code.
6. *f1.number=3;* would be a valid line of code, but *f1.color="blue";* would not be.
7. *f1.price=2.99;* would be a valid line of code, but *w1.price=2.99;* would not be.
8. *e1.check_watermelon(w1, f1);* would be a valid line of code since Watermelon is a derived class from Food.
9. *a1.set_seed();* would be a valid line of code since Apple and Watermelon are derived from the same base class.
10. Because Grape privately inherits from Food, *g1.number=4;* would not be a valid line of code.

Problem 2 (40 points)-Write a program. Submit a file named chili.cpp

Louise is famous in her town for making chili. Every day, she cooks a certain amount (each batch is 4 cups) and she reads in a file with customer information. To save on costs, she uses Starbucks cups (given to her by the local Starbucks) to serve the chili.

Notes:
-You should have a minimum of 2 classes (you decide which classes and how many)

-The program should read in a file in the following format:

Sample file:
Natassa,Grande ***<-Natassa is the customer name, Grande is the Starbucks cup size***
Demy,Tall
Elena,Short

-The program should ask how many batches she will make. If she runs out of chili before all customer orders have been met, there should be an option to make more chili to meet demand

-After an order has been met, the amount of chili left should somehow be indicated. In my sample run, as long as there is more than a certain amount, *Still got lots of chili!* is output to screen. If there is less than this amount, *Not much chili left!* is output

-You should account for at least 4 different Starbucks cup sizes

-There should be an option between orders to take a break or continue. Taking a break can be interpreted as you would like.

Possible Sample Run 1 (your program does not have to match the following exactly, but should have the same functionality):

--Welcome Louise--
Checking today's customers…done!

How many batches of your famous chili are you making today? 1

Starting orders…

--Customer 1: Natassa's order is Grande.
Order served. Still got lots of chili!

Continue with orders or take a break?
continue

--Customer 2: Demy's order is Tall.
Order served. Not much chili left!

Continue with orders or take a break?

break
Ok. Press enter to continue when you are finished with your break.

--Customer 3: Elena's order is Short.
Sorry, not enough chili. Would you like to make another batch or quit?
quit

Problem 3 (40 points)-Write a program. Submit a file named chipotle.cpp

Your friend here in university has an idea to make some extra money. She wants to make a delivery system for Chipotle using students (kept in a file). The program should give users the option to build an order (using actual Chipotle components) and then have a delivery person deliver it. For every order, there is a delivery charge (you decide what this charge is-for example, it can be a percentage of the order).

Notes:
-You should have a minimum of 2 classes (you decide which classes and how many)

-The program should read in a file in the following format:

Sample file:
Serj Tankian ***<-Serj is the delivery person's first name and Tankian is the last name***
Daron Malakian
Shavo Odadjian
John Dolmayan

-The program should give the user the following options:

-Customer (meaning you will put in your order and have it delivered by a delivery person)

-Apply (meaning a person can apply to join as a delivery person). You can do a simple application like my sample run or make it a more complicated process

-Exit

-You can pick any two options from Chipotle to order (I chose a burrito or bowl)

-You should have at least three options for the user to pick from (with each option having at least two items to select-see sample run)

-The total made at the end of the program run (from delivery charges) should be output to screen and a file

Possible Sample Run 1 (your program does not have to match the following exactly, but should have the same functionality):

~~Ronnie's Delivery Service~~

Pick from the following menu:
1. Customer
2. Apply
3. Exit

1

Place your order
Burrito or Bowl?
burrito
Price will be $6.75

-Pick: Tofu, Steak, Chicken
Tofu, Chicken

-Pick: Cilantro-Lime Brown, Cilantro-Lime White
none

-Pick: Queso, Sour Cream, Fresh Tomato Salsa
Queso

Confirm order (yes or no):
Burrito: Tofu, Chicken, Queso
Yes

Ok, Serj will be delivering your order. Thank you.

Pick from the following menu:
1. Customer
2. Apply
3. Exit

2
Enter your full name: Ontronik Khachaturian
Newest delivery person: Ontronik

Pick from the following menu:

1. Customer
2. Apply
3. Exit

3

Exiting…

Total made: ___ //should also be output to file