객체지향프로그래밍

Assignment 4

학과: 수학과

학번: 20140843

이름: 엄태강

POVIS ID: tkeom0114

Honor Code:나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

1. 개요

이번 과제에서는 C++을 사용하여 푸드 트럭 시뮬레이션을 만들어 보았다. 이번 과제에서는 다형성을 이용했다.

2.class 구성

이번 과제에서는 음식에 대한 정보를 가지는 food를 base class로 가지고 그 아래에 다음 그림과 같이 class hierarchy를 구성하였다.

base class인 Food는 아래와 같이 구성되어 있다. 이 class는 abstract base class이다.

class Food{

private:

static int minCount; //최소 선택 횟수를 나타낸다.

static int count; //플레이어가 한 선택 횟수를 나타낸다.

static int correct; //올바르게 조리한 요리의 개수를 나타낸다.

protected:

int needIngredients[7]; //요리에 필요한 재료인지 나타낸다. 1이면 필요한 것이고, 0이면 이미 들어간 재료이고, -1이면 필요 없는 재료이다.

public:

Food (); //Food의 Constructor로 needIngredients를 모두 -1로 초기화 시킨다.

virtual ~Food (void); //Food의 destructor로 모든 재료가 들어간 경우 correct를 1증가시키고, 해당 요리를 제거한다.

virtual void printName (void) = 0; //pure virtual method

void putIngredient (int); //번호에 해당하는 재료를 넣는다. 올바를 재료를 넣은 경우 needIngredients[int]를 0으로 만들고 그렇지 않으면 에러 메시지를 출력한다.

static int getCorrect (void); //correct를 return한다.

static int getCount (void); //count를 return한다.

static int getMinCount (void); //mincount를 return한다.

static void increaseMinCount (void); //mincount를 1증가시킨다.

};

Pizza class는 Food를 상속받아서 아래와 같은 class로 구현했다.

class Pizza: public Food{

public:

Pizza () : Food (){needIngredients[DOUGH] = 1;} //Dough가 필요함

~Pizza (void); //Pizza의 distructor로 mincount를 1증가

virtual void printName (void) = 0; //pure virtual method

};

Coffe class는 Food를 상속받아서 아래와 같은 class로 구현했다.

class Coffee: public Food{

public:

Coffee () : Food (){needIngredients[ESPRESSO] = 1;} //Espresso가 필요함

~Coffee (void); //Coffee의 distructor로 mincount를 1증가

virtual void printName (void) = 0; //pure virtual method

};

Pepperoni class는 Pizza를 상속받아서 아래와 같은 class로 구현했다.

class Pepperoni: public Pizza{

public:

Pepperoni () : Pizza (){needIngredients[PEPPERONI] = 1;} //Pepperoni가 필요함

~Pepperoni (void); // Pepperoni 의 distructor로 mincount를 1증가

virtual void printName (void); //Pepperoni의 이름 출력

};

Potato class는 Pizza를 상속받아서 아래와 같은 class로 구현했다.

class Potato: public Pizza{

public:

Potato () : Pizza (){needIngredients[POTATO] = 1;} //Potato가 필요함

~Potato (void); // Potato 의 distructor로 mincount를 1증가

virtual void printName (void); //Potato의 이름 출력

};

Shrimp class는 Pepperoni를 상속받아서 아래와 같은 class로 구현했다.

class Shrimp: public Pepperoni{

public:

Shrimp () : Pepperoni (){needIngredients[SHRIMP] = 1;} // Shrimp가 필요함

~Shrimp (void); // Shrimp의 distructor로 mincount를 1증가

virtual void printName (void); // Shrimp의 이름 출력

};

Americano class는 Coffee를 상속받아서 아래와 같은 class로 구현했다.

class Americano: public Coffee{

public:

Americano () : Coffee (){needIngredients[WATER] = 1;;} // Water가 필요함

~Americano (void); // Americano의 distructor로 mincount를 1증가

virtual void printName (void); // Americano의 이름 출력

};

Macchiato class는 Coffee를 상속받아서 아래와 같은 class로 구현했다.

class Macchiato: public Coffee{

public:

Macchiato () : Coffee (){needIngredients[MILKFOAM] = 1;;} // Milk Foam이 필요함

~Macchiato (void); // Macchiato의 distructor로 mincount를 1증가

virtual void printName (void); // Macchiato의 이름 출력

};

3.알고리즘

1) main

①요리할 음식들의 list인 list <Food\*> listOfFoods를 만들고, srand(time(0))로 rand함수의 seed를 준다.

②음식을 정할 모드를 선택한다.

③1을 선택하면 정해진 순서대로, 2를 선택하면 랜덤으로 해당하는 요리의 class를 만들어 listOfFood에 넣는다.

④이 때부터 시간을 재고,listOfFoods가 빌 때 까지 printMenu를 호출해 시뮬레이션을 진행한다.

⑤결과를 출력한다.

⑥

⑦

2)시뮬레이션 진행(void printMenu(list<Food\*>&listOfFoods))

①남아있는 요리들을 보여주고, 재료를 선택할 수 있는 메뉴를 출력한다.

②재료를(1~7) 선택하면 (\*listOfFoods.begin())->putIngredient(selectMenu-1)로 선택한 재료를 리스트 맨 앞의 요리에 넣는다. 성공, 실패 여부에 따라 메시지를 출력하고 다시 재료를 선택할 수 있게 한다.

③8번을 선택하면 destructor를 호출하고, list에서 제거한다.

3)재료 추가(void Food::putIngredient(int ingredient))

①ingredient가 base(dough나 espresso)인 경우 둘 중 하나라도 0이라면 이미 있다는 메시지를 출력한다.

②ingredient가 base가 아니고, base가 아직 1이면 base를 먼저 넣어야 한다는 메시지를 출력한다.

③ ①②에 해당하지 않고, needIngredients[gradient]==-1이면 잘못 넣었다는 메시지를 출력한다.

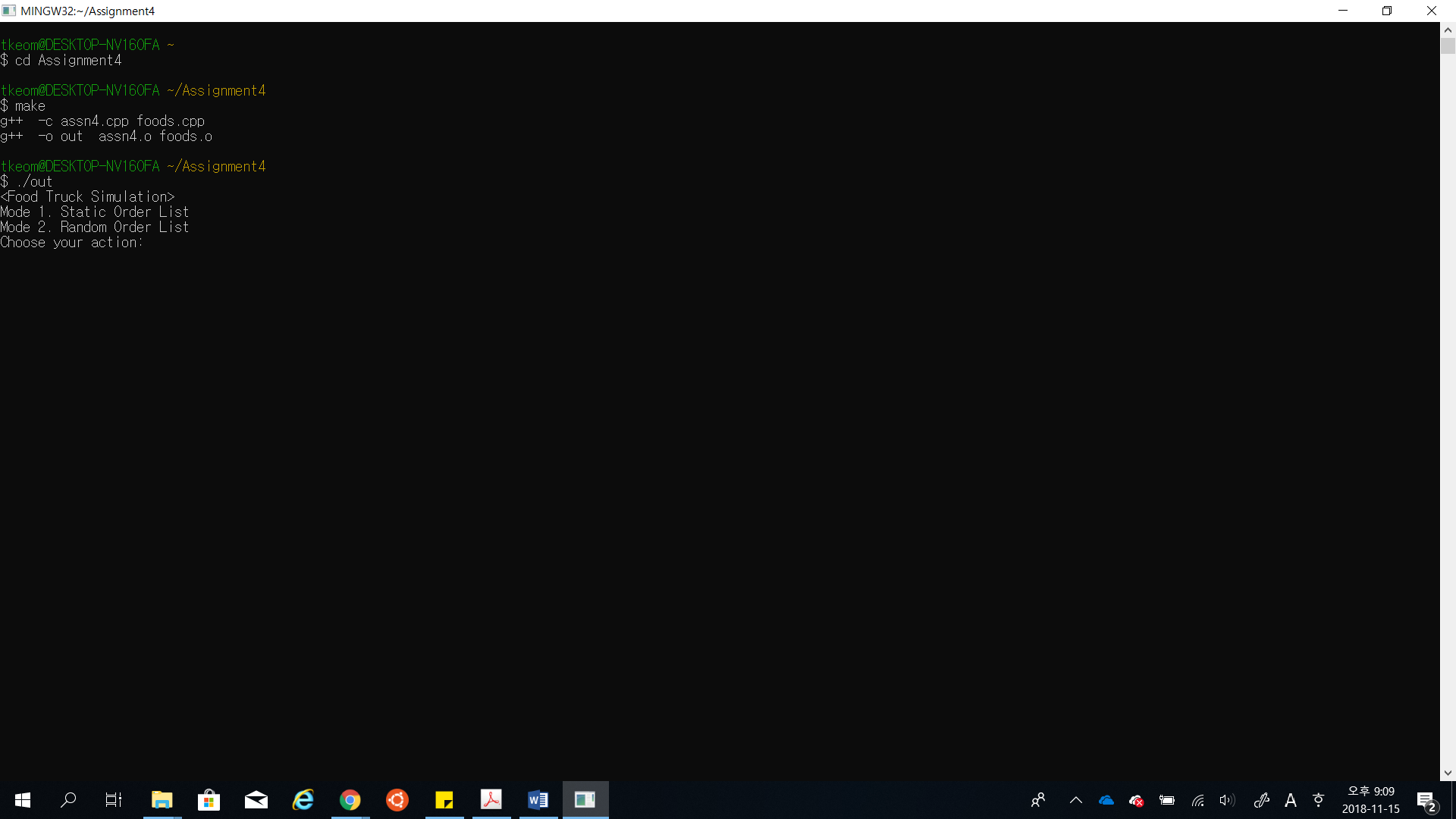
④needIngredients[gradient]==1이면 0으로 만들고 성공 메시지를 출력한다.

4)음식 완성(destructor)

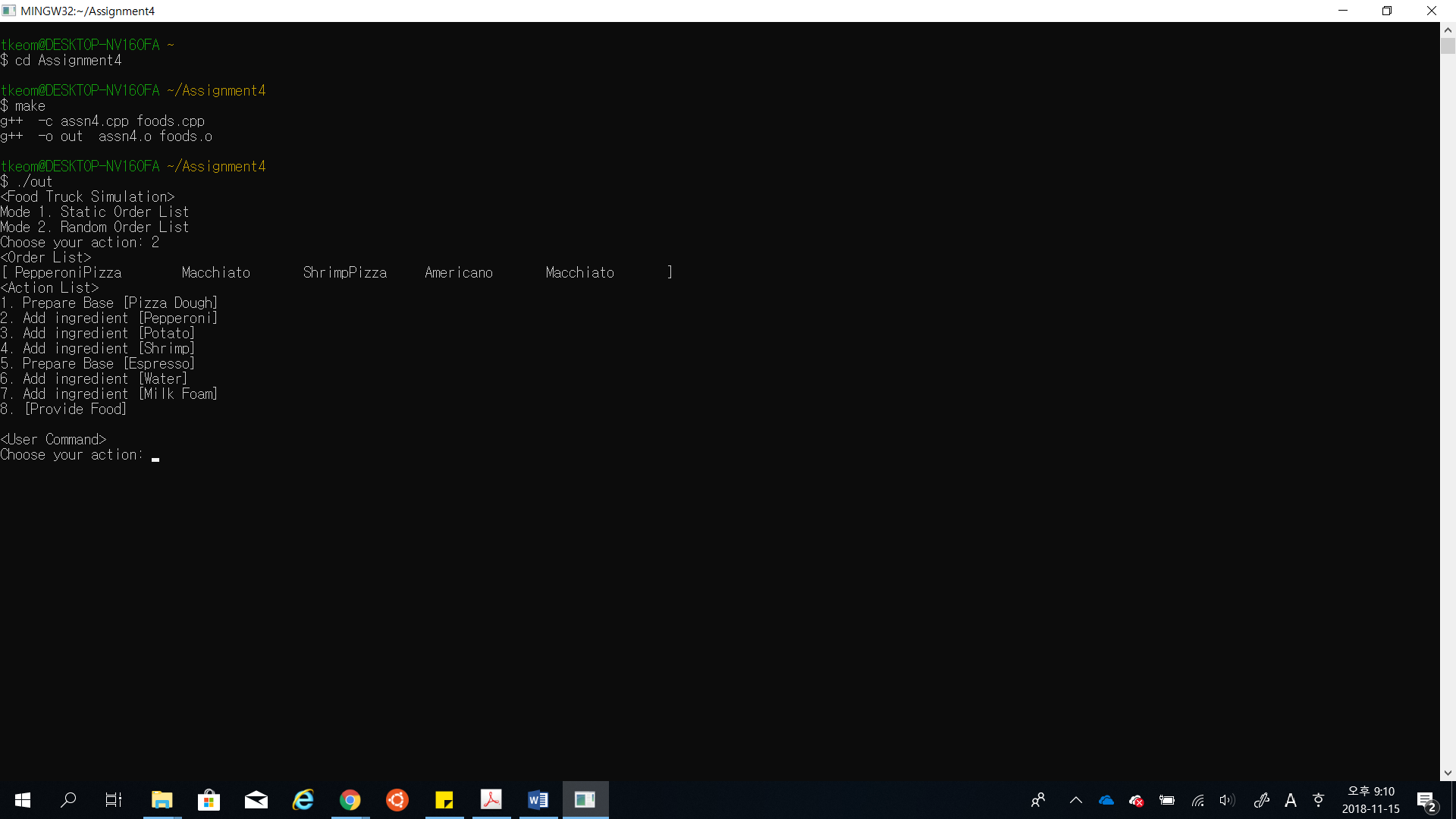
mincount를 이 때 계산하게 되는데 각 class에서 derive하면서 추가된 재료만큼 mincount를 올려 총 필요한 재료들의 개수를 구하게 된다.

4.실행방법&실행예시

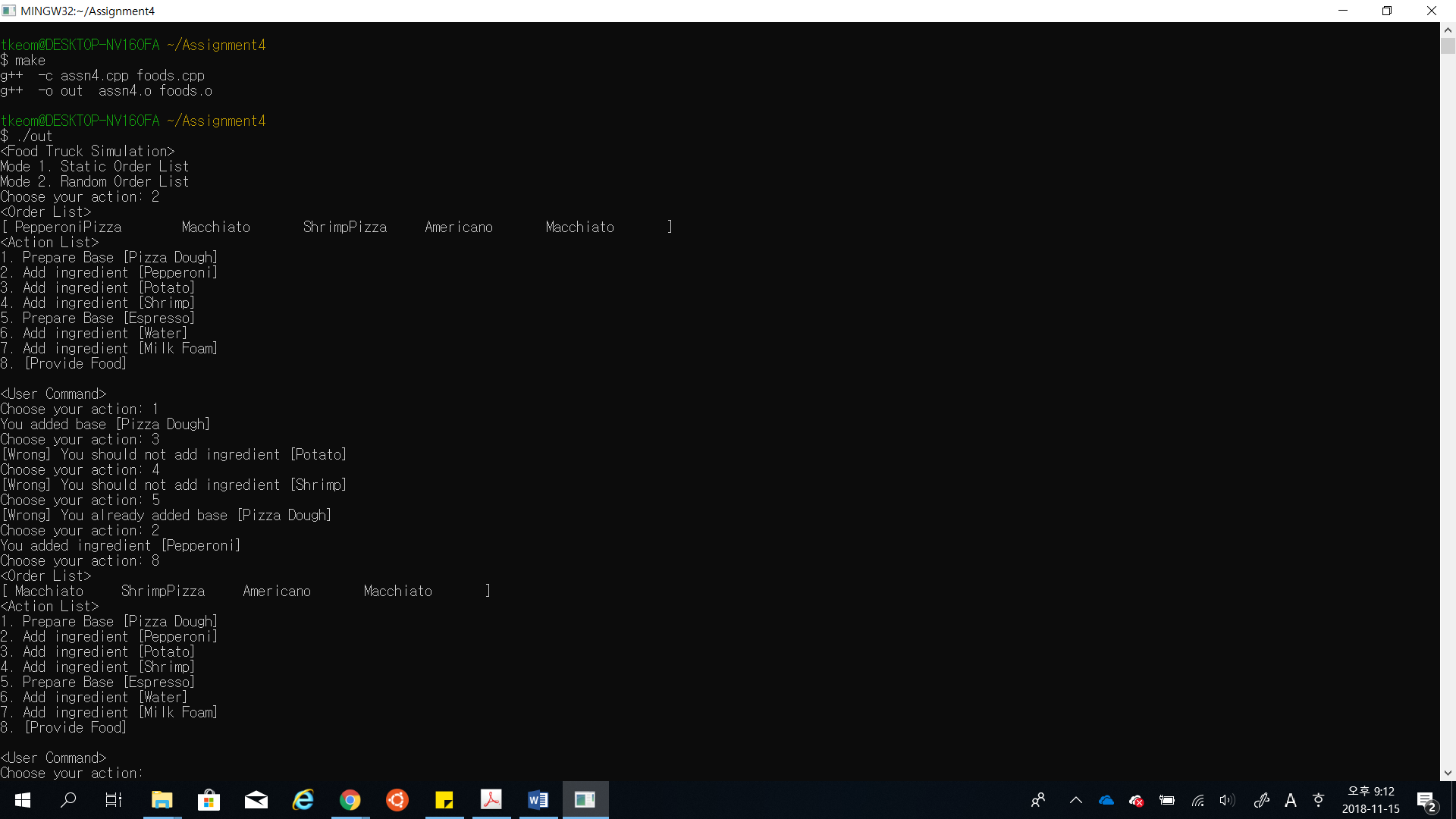
make 후 out파일을 아래와 같이 모드를 선택하는 화면이 출력된다.



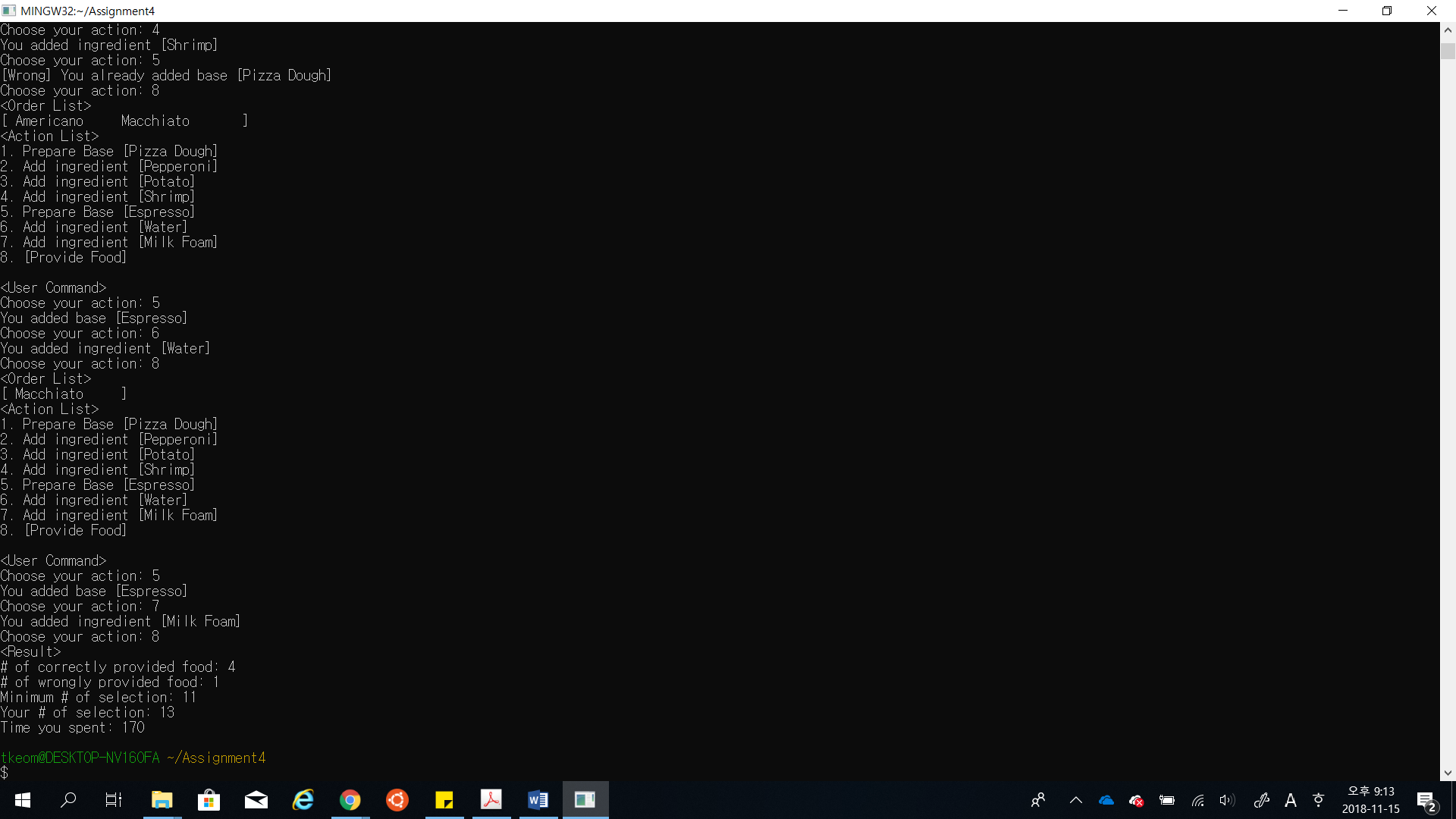
모드를 선택하면 아래와 같이 시뮬레이션이 시작된다.



잘못된 선택을 한 경우 아래와 같이 에러 메시지가 출력되며, 8을 선택하면 list에서 음식에 제거됨도 알 수 있다.



게임이 끝나면 아래와 같이 결과가 출력된다.



5.토론

사실 이번 과제의 경우 Food의 list를 관리하는 과정에서 다형성은 사용했고 실제로 쓰지 않는 가상의 class인 Food를 base로 두었지만 이를 abstract base class로 만드는 방법이 잘 생각나지 않았고, 결국 print함수를 pure virtual method로 두어 해결했다.

6.결론

이번 과제에서는 C++을 사용하여 푸드 트럭 시뮬레이션을 구현했다. 이번 프로젝트에서는 특히 객체지향 프로그래밍의 상속과 다형성을 이용해 프로그램을 작성해 보았다.