menuitem.h

```
#ifndef _MENU_ITEM_H
#define _MENU_ITEM_H
#include <iostream>
#include <string>
using namespace std;

class MenuItem{
 public:
       MenuItem();
       MenuItem(const string &aName, const double &aPrice);
       ~MenuItem(){cout << "DSTR: MenuItem " << name << "\n";}
       void setName(const string &aName){name = aName;}
       void setPrice(const double &aPrice){price = aPrice;};
       string getName(){return name;}
       double getPrice() {return price;}
 private:
       string name;
       double price;
 };

#endif
```

menuitem.cpp

```
#include <iostream>
#include "menuitem.h"
using namespace std;

MenuItem::MenuItem(){
       name="X"; price=-1; cout << "MenuItem cstr\n";
}

MenuItem::MenuItem(const string &aName, const double &aPrice){
       name=aName; price=aPrice;
       cout << "MenuItem paramet. cstr: " << name << " for " << price<<endl;
}
```

orderitem.h

```
#ifndef _ORDER_ITEM_H
#define _ORDER_ITEM_H

#include <iostream>
#include <string>
#include "menuitem.h"
using namespace std;

class OrderItem{
 public:
       OrderItem(){dish=NULL; qty = -1;}
       OrderItem(const MenuItem * aMI, const int &aQty);
       ~OrderItem(){cout << "DSTR: OrderItem "<<getName()<<"\n";}
       string getName(){if (dish !=NULL) return dish->getName(); else return "";}
       double getPrice(){if (dish !=NULL) return dish->getPrice(); else return -1;}
       double getCost(){if (dish !=NULL) return qty * dish->getPrice(); else return -1;}
       int getQty(){return qty;}
       void setQty(const int &aQty){qty = aQty;}
       void setDish(const MenuItem * aMI){dish = const_cast<MenuItem *>(aMI);}
 private:
       MenuItem * dish;
       int qty;
 };

#endif
```

orderitem.cpp

```
#include "orderitem.h"
using namespace std;

OrderItem::OrderItem(const MenuItem * aMI, const int &aQty){
       dish = const_cast<MenuItem *>(aMI); qty = aQty;
}
```


menu.h

```
#ifndef _MENU
#define _MENU

#include "menuitem.h"
using namespace std;

class Menu{
 public:
       Menu(const int &maxNumDishes);
       ~Menu();
       void showMenu();
       MenuItem * const getOfferedDishes() const {return offeredDishes;}
       MenuItem * const getOfferedDishes(const int & pos) const {return &offeredDishes[pos];}
 private:
       MenuItem * offeredDishes;
       int numDishes;
};

#endif
```

menu.cpp

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include "menu.h"
using namespace std;

Menu::Menu(const int &maxNumDishes){
       offeredDishes = new MenuItem[maxNumDishes];
       ifstream in("dishes.ascii");
       if(in.good()!= true) exit(EXIT_FAILURE);
       int i =0;
       while ((in.eof() != true) && (i <= maxNumDishes)){
              string aName; float aPrice;
              in >> aName    >> aPrice;
              offeredDishes[i].setName(aName); offeredDishes[i].setPrice(aPrice);
              i++;
       }
       in.close();
       numDishes = i;
       cout << "MENU cstr with " << numDishes << " dishes \n";
}

Menu::~Menu(){
       delete [] offeredDishes;
       cout << "DSTR: Menu\n";
}

void Menu::showMenu(){
       cout << "---------------- MENU -------------------- \n";
       for (int i = 0; i < numDishes; i++){
                     cout <offeredDishes[i].getName()<< "\t"<<offeredDishes[i].getPrice()<<endl;
       }
       cout << "---------------- END MENU -------------------- \n";
}
```

order.h

```
#ifndef _ORDER_H
#define _ORDER_H

#include <iostream>
#include "orderitem.h"
#include "menu.h"
using namespace std;

class Order{
 public:
       Order(const int &tNum, const int &orderSize, const int & nPersons, const Menu & m);
       ~Order();
       void showOrder();
       void computeDetails(double * avgCostPerPerson, double * totalOrderCost);
 private:
       OrderItem * items;
       int tableNumber;
       int countItems;
       int numPersons;
 };


#endif
```

order.cpp

```
#include <iostream>
#include "order.h"
using namespace std;


Order::Order(const int &tNum, const int &orderSize, const int & nPersons, const Menu & m):
tableNumber(tNum), countItems(orderSize){
       //a little bit faster than saying tableNum = tNum; countItems = orderSize;
       numPersons = nPersons;       //better to add in the above list: numPersons(nPersons)
       items = new OrderItem[countItems];
       for (int i =0; i< countItems; i++){
              int itemPos; int aQty;
              cout << "Give item's position in Menu and quantity: "; cin >> itemPos >> aQty;
              items[i].setQty(aQty); items[i].setDish(m.getOfferedDishes(itemPos));
       }
}

Order::~Order(){
       delete [] items;
       cout << "DSTR: Order\n";
}

void Order::computeDetails(double * avgCostPerPerson, double * totalOrderCost){
       double totalCost = 0;
       for (int i =0; i< countItems; i++){
              totalCost += items[i].getCost();
       }
       *totalOrderCost = totalCost;
       *avgCostPerPerson = totalCost / numPersons;
}

void Order::showOrder(){
       cout << "\n---------- ORDER details ----------\n";
       for (int i =0; i< countItems; i++){
          cout <<items[i].getQty()<<" x "<<items[i].getName()<<" for "<< items[i].getCost()<<endl;
       }
       double avgCostPerPerson; double totalCost;
       computeDetails(&avgCostPerPerson, &totalCost);
       cout << "----------------------------\n";
       cout <<"Total cost is "<<totalCost<<" with average cost "<<avgCostPerPerson<<" per head\n";
       cout << "---------- END ORDER details ----------\n";
}
```


restaurant.cpp

```
#include <iostream>
#include <cstdlib>

#include "menu.h"
#include "order.h"
using namespace std;

int main(){

       Menu m(12);
       m.showMenu();
       Order * o;

       int exitFlag = 1;
       while(exitFlag){
              cout << "Give 0 for exit, 1 for new order, 2 for cancellation: ";
              int choice;
              cin >> choice;
              switch(choice){
              case 0: cout << "\nExiting. Na mas 3anar6ete...\n";
                     //exit(EXIT_SUCCESS);
                     exitFlag = 0;  //observe how these alternatives affect dest's!
                     break;
              case 1: cout << "Please give the order size for table 14 (4 persons):\n";
                     int oSize;
                     cin >> oSize;
                     o = new Order(14, oSize, 4, m);
                     o->showOrder();
                     break;
              case 2: cout << "\n ---------- KILLED BY DEATH ---------- \n";
                     cout << "Gonna delete the order o, see who dies\n";
                     delete o;
                     cout << "\n ---------- KILLED BY DEATH ---------- \n";
                     break;
              default:       cout <<"Wrong choice, try again\n";
                     break;
              }
       }
       return 0;
}
```

makefile

```
CC = g++
OPT = -g -Wall

OBJ = ./obj/order.o \
       ./obj/orderitem.o \
       ./obj/menu.o \
       ./obj/menuitem.o
EXE= ektoraSoseMas.exe

all: $(EXE)

$(EXE):./src/restaurant.cpp $(OBJ)
       $(CC) $(OPT) -o $(EXE) ./src/restaurant.cpp $(OBJ)

./obj/order.o: ./src/order.h ./src/order.cpp ./src/orderitem.h ./src/menu.h
       $(CC) $(OPT) -c  -o ./obj/order.o ./src/order.cpp

./obj/orderitem.o: ./src/orderitem.h ./src/orderitem.cpp ./src/menuitem.h
       $(CC) $(OPT) -c  -o ./obj/orderitem.o ./src/orderitem.cpp

./obj/menu.o: ./src/menu.h ./src/menu.cpp ./src/menuitem.h
       $(CC) $(OPT) -c -o ./obj/menu.o ./src/menu.cpp

./obj/menuitem.o: ./src/menuitem.h ./src/menuitem.cpp
       $(CC) $(OPT) -c -o ./obj/menuitem.o  ./src/menuitem.cpp

clean:
       rm $(OBJ) $(EXE)
```