```
//
//  main.cpp
//  Shooping List
//
//  Created by daylin on 22/10/2015.
//  Copyright (c) 2015 daylin. All rights reserved.
//

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


struct node
{

    string name;
    unsigned int quant;
    double price;
    node*next;

};



node * head=0;
node * tail=0;
node * curr=0;


void name_sorting ()
{
    node * temphead = head;
    string tempname;
    int counter = 0;
    while (temphead)
    {
        temphead = temphead->next;
        counter++;
    }
    temphead = head;

    for (int j=0; j<counter; j++)
    {
        while (temphead->next)
        {
            if (temphead->name > temphead->next->name)
            {
                tempname = temphead->name;
                temphead->name = temphead->next->name;
                temphead->next->name = tempname;
            }
            temphead = temphead->next;
        }
        temphead = head;
    }
}
//void print;
void print(int num, int T)
{
    unsigned int n=1;
    node *curr = head;

    cout<< "Your list contains "<< num << " item for a total of "<< T
<<"EUR:"<<endl;
    while (curr!=NULL)
    {

        cout << n++ << ") "<< curr->name<<": "<< curr->quant<<" x "<< curr-
>price<<" EUR. Total: "<< curr->quant*curr->price<<endl;
        curr = curr->next;

    }
    cout << endl;
}

void addToTail(int& n, double& T,string name,unsigned int quant,double
price)
{
    n=n+1;
    T = T+(quant*price);
    // Create new node
    node *newnode = new node;

    newnode->name = name;
    newnode->quant= quant;
    newnode->price= price;

    newnode->next = NULL;
    // add it to the tail of the list
    if (tail == NULL)
        {
```

```
        // list was empty
        head = tail = newnode;
        }
    else {
            tail->next = newnode;
            tail = newnode;
        }
}

/**
 * Find and remove the first occurrence of data
 * Return true iif the data was found and removed
 */
bool remove(int& n, double& T,string name)
{

    n=n-1;
    node *curr = head;
    node *prev = NULL;
    T=T-(curr->price*curr->quant);
    // iterate until end of the list
    while (curr!=NULL) {
        // compare the current data with the one to delete
        if (curr->name == name)
        { // found match!
            // Is it the first element?
            if (curr == head)
            { // special case: head
                head = curr->next; // update head to the next
            }
            else
                { // any other case: perform bypass
                prev->next = curr->next; //bypass
                }

            if (curr == tail)
                {// special case: tail
                    tail = prev; // next should be the new tail
                }
            delete curr; // delete the node
            return true; // job done!
        }
        // move to the next one
        prev = curr; // remember the previous
        curr = curr->next;
    }
    return false;
}

/**
 * Clear the list by deleting all items
 */
void destroy()
{
    node *next, *curr = head;
    // iterate until end of the list
    while (curr!=NULL)
    {
        next = curr->next; // remember the next
        delete curr;       // delete current
        curr = next;       // move to next
    }
}



int main()
{
    int num=0; // stores the number of items on the list
    double total=0;// stores the total value os the items On the
declarations of the functions I called it "T"
    int choice=0;
    node node;


    cout <<"Your list is empty."<<endl;
    cout <<"Make a choice: (1) Add item, (2) Remove item, (3) Exit "<<endl;
    cin >> choice;
    while (choice!=3)
    {
        switch (choice)
        {
        case 1:
            cout<< "Enter new item: ";
            cin >> node.name;
            cout<< "Enter quantity: ";
            cin >> node.quant;
            cout<< "Enter unit price (EUR): ";
            cin >> node.price;
            cout<<endl;
            addToTail( num,total,node.name,node.quant,node.price);
            name_sorting ();
            print(num, total);

            cout <<"Make a choice: (1) Add item, (2) Remove item, (3) Exit "<<endl;
            break;

        case 2:
            cout<<"Which item do you want to delete? "<<endl;
            cin >>node.name;
            remove(num,total,node.name);

            name_sorting ();
            print(num, total);
            cout <<"Make a choice: (1) Add item, (2) Remove item, (3) Exit "<<endl;
            break;

        default:
            cout <<"Your list is empty."<<endl;
            cout <<"Make a choice: (1) Add item, (2) Remove item, (3) Exit "<<endl;
                break;
        }

        cin>>choice;

    }
    if (choice==3)
    cout <<"Bye!!"<<endl;
    destroy();
    return 0;
}
```