```cpp
#include <iostream>
#include <string.h>
using  namespace  std  ;

struct  produit
{
    char      code[9]    ;
    char      name[50]   ;
    char      MU[3]      ;
    int       Qty        ;
    double    PET        ;
    double    VAT        ;

    struct produit *  next  ;
};

typedef  struct  produit  Element  ;
typedef  Element *        List     ;

List  create_node()
{
    List  ptr  =  new  Element  ;

    cout  <<  "Give the product code : "  ;
    cin   >>  ptr->code                   ;

    cout  <<  "Give the product name : "  ;
    cin   >>  ptr->name                   ;
    cout  <<  "Give the product MU : "    ;
    cin   >>  ptr->MU                     ;
    cout  <<  "Give the product quantity : "  ;
    cin   >>  ptr->Qty                    ;
    cout  <<  "Give the product PET : "   ;
    cin   >>  ptr->PET                    ;
    cout  <<  "Give the product VAT : "   ;
    cin   >>  ptr->VAT                    ;

    ptr->next  =  NULL  ;

    return  ptr ;
}

void  display_list(List lst)
{
    List  ptr = lst ;

    if(lst == NULL)
    {
        cout  <<  "The product list is empty."  <<  endl  <<  endl ;
        return  ;
    }

    while(ptr != NULL)
    {
        cout  <<  "The product code is : "  <<  ptr->code  <<  endl ;
        cout  <<  "The product name is : "  <<  ptr->name  <<  endl ;
        cout  <<  "The product MU is : "    <<  ptr->MU  <<  endl ;
        cout  <<  "The product quantity is : "  <<  ptr->Qty  <<  endl ;
        cout  <<  "The product PET is : "   <<  ptr->PET  <<  endl ;
        cout  <<  "The product VAT is : "   <<  ptr->VAT  <<  endl  <<  endl  ;

        ptr  =  ptr->next  ;
    }

    return  ;
}

List  add_node(List list)
{
    List  ptr  =  create_node()  ;
    ptr->next  =  list  ;
    list       =  ptr   ;

    return  list  ;
}
```

```
List  delete_node(List lst, char new_code[])
{
    if(lst != NULL)
    {
        List  ptr  =  lst  ;

        while((ptr != NULL)&&(strcmp(ptr->code, new_code) != 0))
            ptr  =  ptr->next  ;

        if((ptr != NULL)&&(strcmp(ptr->code, new_code) == 0))
        {
            if(ptr == lst)
            {
                lst  =  lst->next  ;
                delete       ptr   ;

                return       lst   ;
            }
            else
            {
                List  prv  =  lst  ;

                while(prv->next != ptr)
                    prv  =  prv->next  ;

                prv->next  =  ptr->next  ;
                delete        ptr        ;

                return  lst  ;
            }
        }
        else
            return  lst  ;
    }
    else
        return  NULL ;
}

List  empting_list(List lst)
{
    List  ptr  ;

    while(lst != NULL)
    {
        ptr = lst         ;
        lst = lst->next   ;
        delete  ptr       ;
    }

    return  lst  ;
}

void  min_max(List lst)
{
    List  ptr  =  lst  ;

    double  min = 0.0, max = 0.0 ;

    if(lst != NULL)
    {
        min  =  lst->PET  ;
        max  =  lst->PET  ;
    }

    while(ptr != NULL)
    {
        if(ptr->PET < min)
        {
            min  =  ptr->PET  ;
        }

        if(ptr->PET > max)
        {
            max  =  ptr->PET  ;
        }

        ptr  =  ptr->next  ;
```

```
    }

    cout  <<  "Minimum price = "  <<  min  <<  endl  ;
    cout  <<  "Maximum price = "  <<  max  <<  endl  ;

    return ;
}

int  nodes_number(List lst)
{
    List ptr     =  lst  ;
    int   number =  0    ;

    while(ptr != NULL)
    {
        ptr  =  ptr->next  ;
        number++           ;
    }

    return  number  ;
}

double  total_PET(List lst)
{
    List    ptr  =  lst  ;
    double  pet  =  0.0  ;

    while(ptr != NULL)
    {
        pet  +=  (ptr->PET * ptr->Qty)    ;
        ptr   =  ptr->next  ;
    }

    return  pet  ;
}

double  total_PIT(List lst)
{
    List    ptr  =  lst  ;
    double  pwt  =  0.0  ;

    while(ptr != NULL)
    {
        pwt  +=  ((ptr->PET + (ptr->PET * ptr->VAT)) * ptr->Qty)   ;
        ptr   =  ptr->next  ;
    }

    return  pwt  ;
}

int  main()
{
    List  lst  =  NULL ;

    int option  =  0  ;

    cout  <<  "CASHIER PROGRAM"  <<  endl  ;

    do
    {
        cout  <<  endl  <<  "you have many options :"  <<  endl  ;
        cout  <<  "1. add a product."  <<  endl  ;
        cout  <<  "2. display all bought products."  <<  endl  ;
        cout  <<  "3. display the number of bought products, their price without taxes and price with taxes, and the taxes amount."  <<  endl  ;
        cout  <<  "4. checkout, and get the taxes amount."  <<  endl  ;
        cout  <<  "5. delete a product."  <<  endl  ;
        cout  <<  "6. display the minimum and the maximum price."  <<  endl  ;
        cout  <<  "7. empty the product list."  <<  endl  ;
        cout  <<  "8. end the program."  <<  endl  <<  endl  ;
        cout  <<  "Enter the corresponding option number :"  ;

        cin  >>  option  ;

        switch(option)
        {
```

```cpp
            case  1 :
                cout  <<  "add information of the new product :"  <<  endl  ;
                lst  =  add_node(lst)  ;
                break  ;

            case  2 :
                cout  <<  "display all bought products :"  <<  endl  <<  endl  ;
                display_list(lst)  ;
                break  ;

            case  3 :
                cout  <<  "the number of bought products is : "  <<  nodes_number(lst)  <<endl;
                cout  <<  "the products price without taxes : "  <<  total_PET(lst)     <<endl;
                cout  <<  "the products price with taxes : "     <<  total_PIT(lst)     <<endl;
                cout  <<  "the taxes amount : "  <<  total_PIT(lst) - total_PET(lst)    <<endl;
                break  ;

            case  4 :
                cout  <<  "the total price is : "  <<  total_PIT(lst)  <<  endl  ;
                cout  <<  "the taxes amount : "    <<  total_PIT(lst) - total_PET(lst) << endl;
                lst  =  empting_list(lst)  ;
                break  ;

            case  5 :
                char  code[9]  ;
                cout  <<  "Give the code of the product to delete : "  ;
                cin   >>  code   ;
                lst  =  delete_node(lst, code)  ;
                break  ;

            case  6 :
                min_max(lst)  ;
                break  ;

            case  7 :
                lst  =  empting_list(lst)  ;
                break  ;

            case  8 :
                lst  =  empting_list(lst)  ;
                break  ;

            default :
                lst  =  empting_list(lst)  ;
                return   0         ;
        }
    }
    while(option != 8);

    return  0  ;
}
```