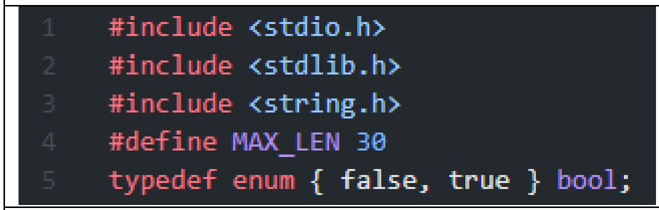
**MARMARA UNIVERSITY COMPUTER ENGINEERING**
**CSE2225 DATA STRUCTURES**
**REPORT OF PROJECT 1**
**APRIL 2021**

| | |
|---|---|
| **PROFESSOR** | MURAT CAN GANİZ |

**GROUP MEMBERS**

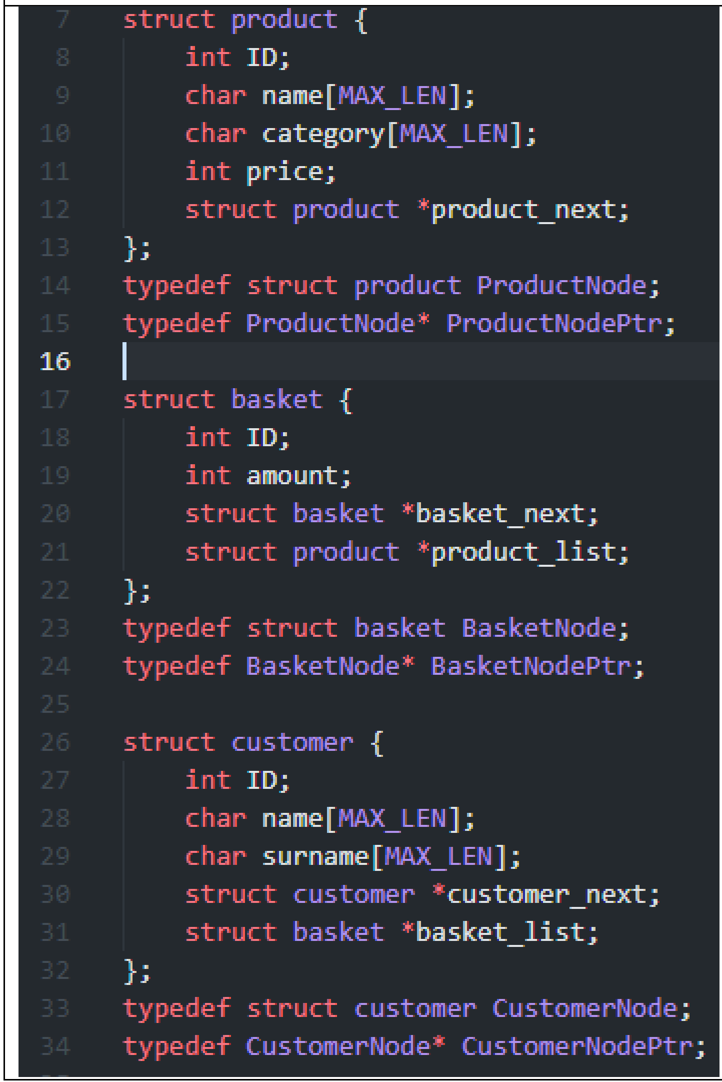
| | | |
|---|---|---|
| ELİF NUR | KEMİKSİZ | 100217006 |
| NESRİN | ŞİMŞEK | 150119664 |
| REYTA GÜL | MURAN | 150117028 |

## INTRODUCTION



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_LEN 30
typedef enum { false, true } bool;
```

## STRUCTURES

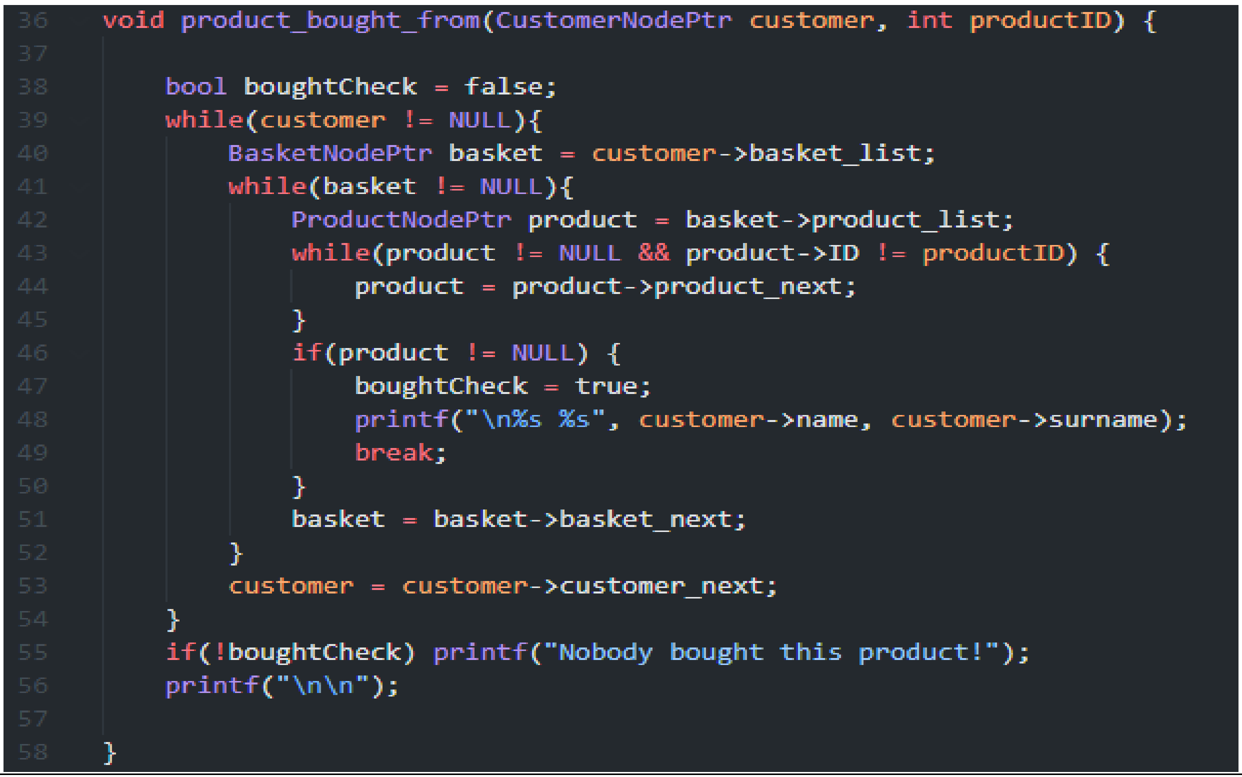


```c
struct product {
    int ID;
    char name[MAX_LEN];
    char category[MAX_LEN];
    int price;
    struct product *product_next;
};
typedef struct product ProductNode;
typedef ProductNode* ProductNodePtr;

struct basket {
    int ID;
    int amount;
    struct basket *basket_next;
    struct product *product_list;
};
typedef struct basket BasketNode;
typedef BasketNode* BasketNodePtr;

struct customer {
    int ID;
    char name[MAX_LEN];
    char surname[MAX_LEN];
    struct customer *customer_next;
    struct basket *basket_list;
};
typedef struct customer CustomerNode;
typedef CustomerNode* CustomerNodePtr;
```

- There are 3 structures that hold properties and pointers which points to their next nodes or linked lists derived from other structures.

## FUNCTIONS

### 1. FUNCTION product_bought_from



```c
void product_bought_from(CustomerNodePtr customer, int productID) {

    bool boughtCheck = false;
    while(customer != NULL){
        BasketNodePtr basket = customer->basket_list;
        while(basket != NULL){
            ProductNodePtr product = basket->product_list;
            while(product != NULL && product->ID != productID) {
                product = product->product_next;
            }
            if(product != NULL) {
                boughtCheck = true;
                printf("\n%s %s", customer->name, customer->surname);
                break;
            }
            basket = basket->basket_next;
        }
        customer = customer->customer_next;
    }
    if(!boughtCheck) printf("Nobody bought this product!");
    printf("\n\n");

}
```

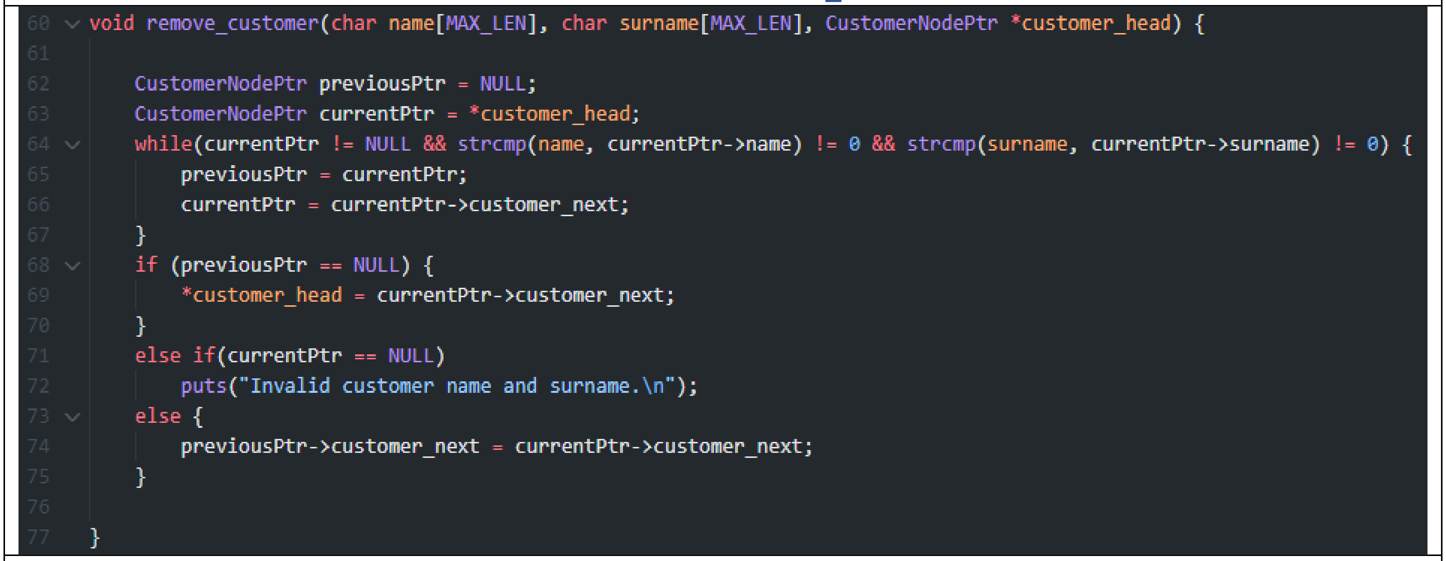
### PARAMETERS

- CustomerNodePtr customer: Takes the start pointer of customers.
- int productID: Takes product ID that is wanted to be searched.

### FUNCTION

- When the user selects 4 from the menu, the function lists the customers who purchased the selected product.
- If the product has not been purchased before, the function gives a warning message.

## 2. FUNCTION remove_customer



```c
void remove_customer(char name[MAX_LEN], char surname[MAX_LEN], CustomerNodePtr *customer_head) {

    CustomerNodePtr previousPtr = NULL;
    CustomerNodePtr currentPtr = *customer_head;
    while(currentPtr != NULL && strcmp(name, currentPtr->name) != 0 && strcmp(surname, currentPtr->surname) != 0) {
        previousPtr = currentPtr;
        currentPtr = currentPtr->customer_next;
    }
    if (previousPtr == NULL) {
        *customer_head = currentPtr->customer_next;
    }
    else if(currentPtr == NULL)
        puts("Invalid customer name and surname.\n");
    else {
        previousPtr->customer_next = currentPtr->customer_next;
    }

}
```

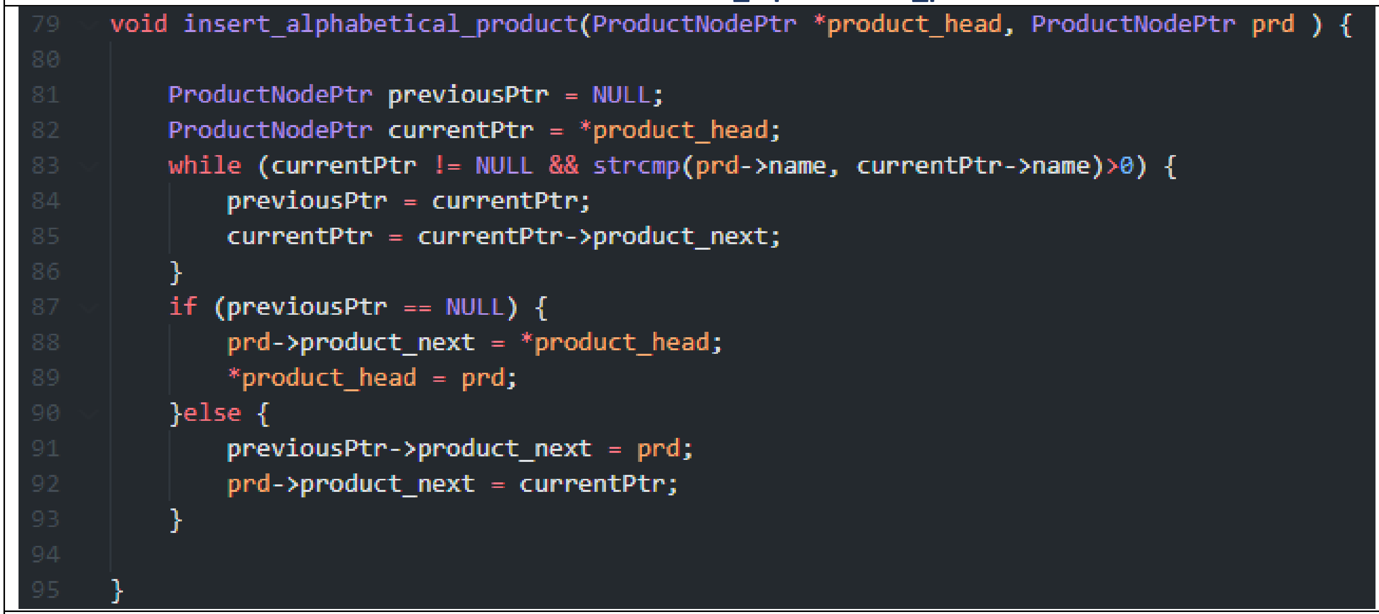
**PARAMETERS**

- char name[MAX_LEN]: Takes customer’s name.
- char surname[MAX_LEN]: Takes customer’s surname.
- CustomerNodePtr *customer_head: Takes the start pointer of customers as an address.

**FUNCTION**

- When the user selects 3 from the menu, removes related customer from customer linked list.
- If there is no such a customer, the function gives a warning message.

## 3. FUNCTION insert_alphabetical_product



```c
void insert_alphabetical_product(ProductNodePtr *product_head, ProductNodePtr prd ) {

    ProductNodePtr previousPtr = NULL;
    ProductNodePtr currentPtr = *product_head;
    while (currentPtr != NULL && strcmp(prd->name, currentPtr->name)>0) {
        previousPtr = currentPtr;
        currentPtr = currentPtr->product_next;
    }
    if (previousPtr == NULL) {
        prd->product_next = *product_head;
        *product_head = prd;
    }else {
        previousPtr->product_next = prd;
        prd->product_next = currentPtr;
    }

}
```

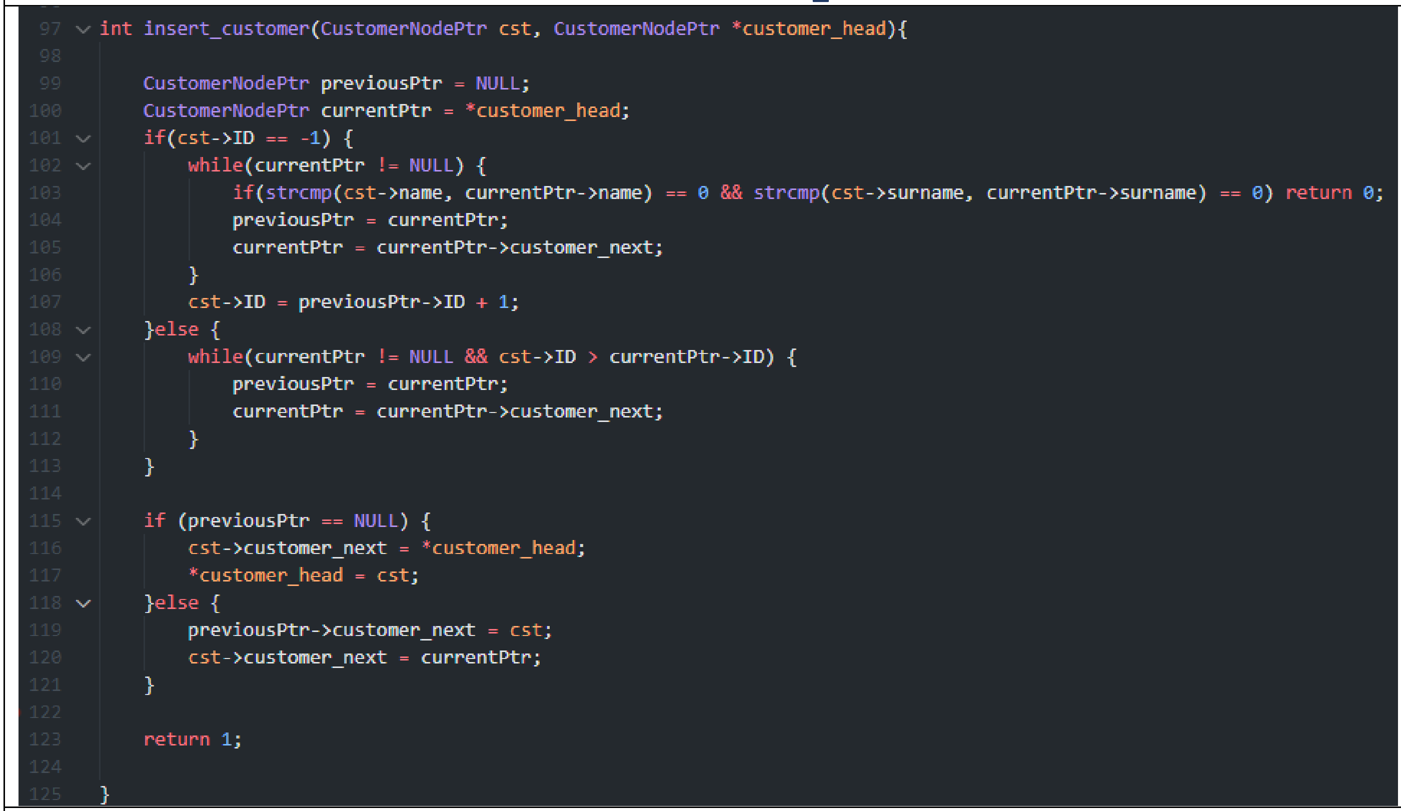
**PARAMETERS**

- ProductNodePtr *product_head: Takes the start pointer of products as an address.
- ProductNodePtr prd: The instance of struct product.

**FUNCTION**

- Alphabetically sorts the products according to their names.

## 4. FUNCTION insert_customer



```c
int insert_customer(CustomerNodePtr cst, CustomerNodePtr *customer_head){

    CustomerNodePtr previousPtr = NULL;
    CustomerNodePtr currentPtr = *customer_head;
    if(cst->ID == -1) {
        while(currentPtr != NULL) {
            if(strcmp(cst->name, currentPtr->name) == 0 && strcmp(cst->surname, currentPtr->surname) == 0) return 0;
            previousPtr = currentPtr;
            currentPtr = currentPtr->customer_next;
        }
        cst->ID = previousPtr->ID + 1;
    }else {
        while(currentPtr != NULL && cst->ID > currentPtr->ID) {
            previousPtr = currentPtr;
            currentPtr = currentPtr->customer_next;
        }
    }

    if (previousPtr == NULL) {
        cst->customer_next = *customer_head;
        *customer_head = cst;
    }else {
        previousPtr->customer_next = cst;
        cst->customer_next = currentPtr;
    }

    return 1;

}
```

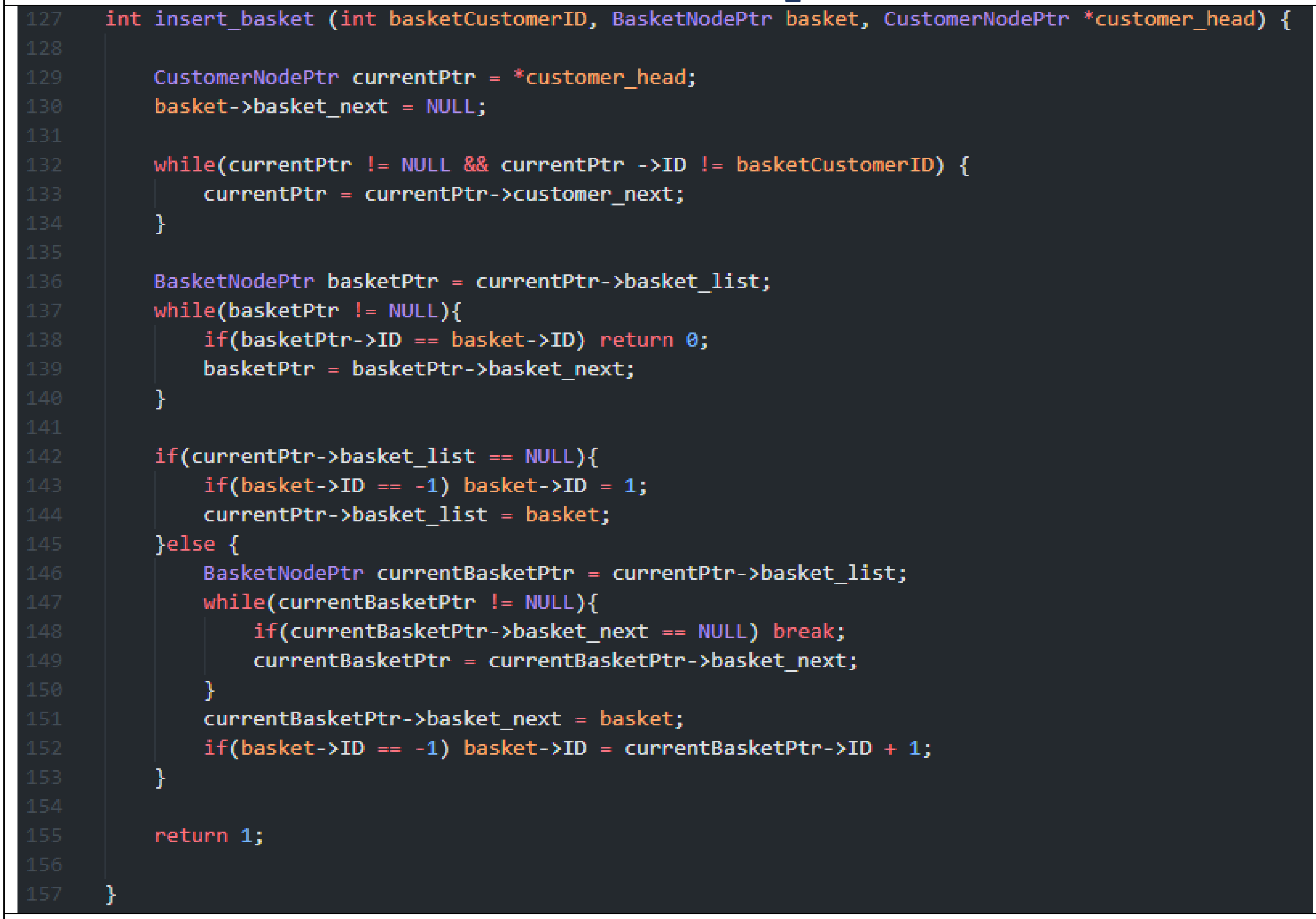
**PARAMETERS:**

- CustomerNodePtr cst: The instance of struct customer.
- CustomerNodePtr *customer_head: Takes the start pointer of customers as an address.

**FUNCTION:**

- When the user selects 1 from the menu, inserts related customer to customer linked list.

## 5. FUNCTION insert_basket



```c
int insert_basket (int basketCustomerID, BasketNodePtr basket, CustomerNodePtr *customer_head) {

    CustomerNodePtr currentPtr = *customer_head;
    basket->basket_next = NULL;

    while(currentPtr != NULL && currentPtr ->ID != basketCustomerID) {
        currentPtr = currentPtr->customer_next;
    }

    BasketNodePtr basketPtr = currentPtr->basket_list;
    while(basketPtr != NULL){
        if(basketPtr->ID == basket->ID) return 0;
        basketPtr = basketPtr->basket_next;
    }

    if(currentPtr->basket_list == NULL){
        if(basket->ID == -1) basket->ID = 1;
        currentPtr->basket_list = basket;
    }else {
        BasketNodePtr currentBasketPtr = currentPtr->basket_list;
        while(currentBasketPtr != NULL){
            if(currentBasketPtr->basket_next == NULL) break;
            currentBasketPtr = currentBasketPtr->basket_next;
        }
        currentBasketPtr->basket_next = basket;
        if(basket->ID == -1) basket->ID = currentBasketPtr->ID + 1;
    }

    return 1;

}
```

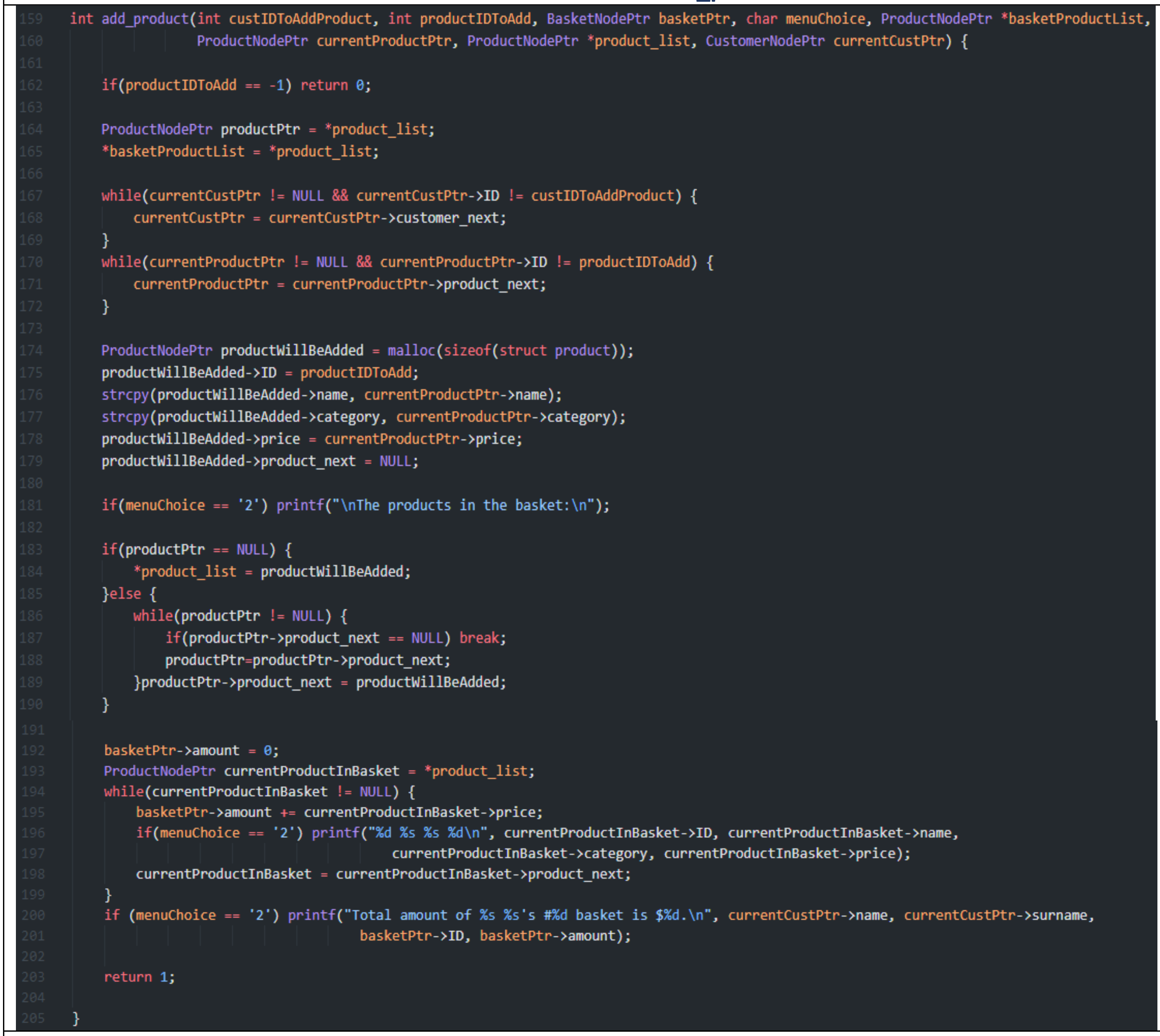
### PARAMETERS

- int basketCustomerID: Takes customer’s ID in the basket.txt.
- BasketNodePtr basket: The instance of struct basket.
- CustomerNodePtr *customer_head: Takes start pointer of customers as an address.

### FUNCTION

- It inserts baskets according to basket.txt and the data entered by the user.

## 6. FUNCTION add_product



```c
int add_product(int custIDToAddProduct, int productIDToAdd, BasketNodePtr basketPtr, char menuChoice, ProductNodePtr *basketProductList,
                ProductNodePtr currentProductPtr, ProductNodePtr *product_list, CustomerNodePtr currentCustPtr) {

    if(productIDToAdd == -1) return 0;

    ProductNodePtr productPtr = *product_list;
    *basketProductList = *product_list;

    while(currentCustPtr != NULL && currentCustPtr->ID != custIDToAddProduct) {
        currentCustPtr = currentCustPtr->customer_next;
    }
    while(currentProductPtr != NULL && currentProductPtr->ID != productIDToAdd) {
        currentProductPtr = currentProductPtr->product_next;
    }

    ProductNodePtr productWillBeAdded = malloc(sizeof(struct product));
    productWillBeAdded->ID = productIDToAdd;
    strcpy(productWillBeAdded->name, currentProductPtr->name);
    strcpy(productWillBeAdded->category, currentProductPtr->category);
    productWillBeAdded->price = currentProductPtr->price;
    productWillBeAdded->product_next = NULL;

    if(menuChoice == '2') printf("\nThe products in the basket:\n");

    if(productPtr == NULL) {
        *product_list = productWillBeAdded;
    }else {
        while(productPtr != NULL) {
            if(productPtr->product_next == NULL) break;
            productPtr=productPtr->product_next;
        }productPtr->product_next = productWillBeAdded;
    }

    basketPtr->amount = 0;
    ProductNodePtr currentProductInBasket = *product_list;
    while(currentProductInBasket != NULL) {
        basketPtr->amount += currentProductInBasket->price;
        if(menuChoice == '2') printf("%d %s %s %d\n", currentProductInBasket->ID, currentProductInBasket->name,
                                     currentProductInBasket->category, currentProductInBasket->price);
        currentProductInBasket = currentProductInBasket->product_next;
    }
    if (menuChoice == '2') printf("Total amount of %s %s's #%d basket is $%d.\n", currentCustPtr->name, currentCustPtr->surname,
                                  basketPtr->ID, basketPtr->amount);

    return 1;

}
```

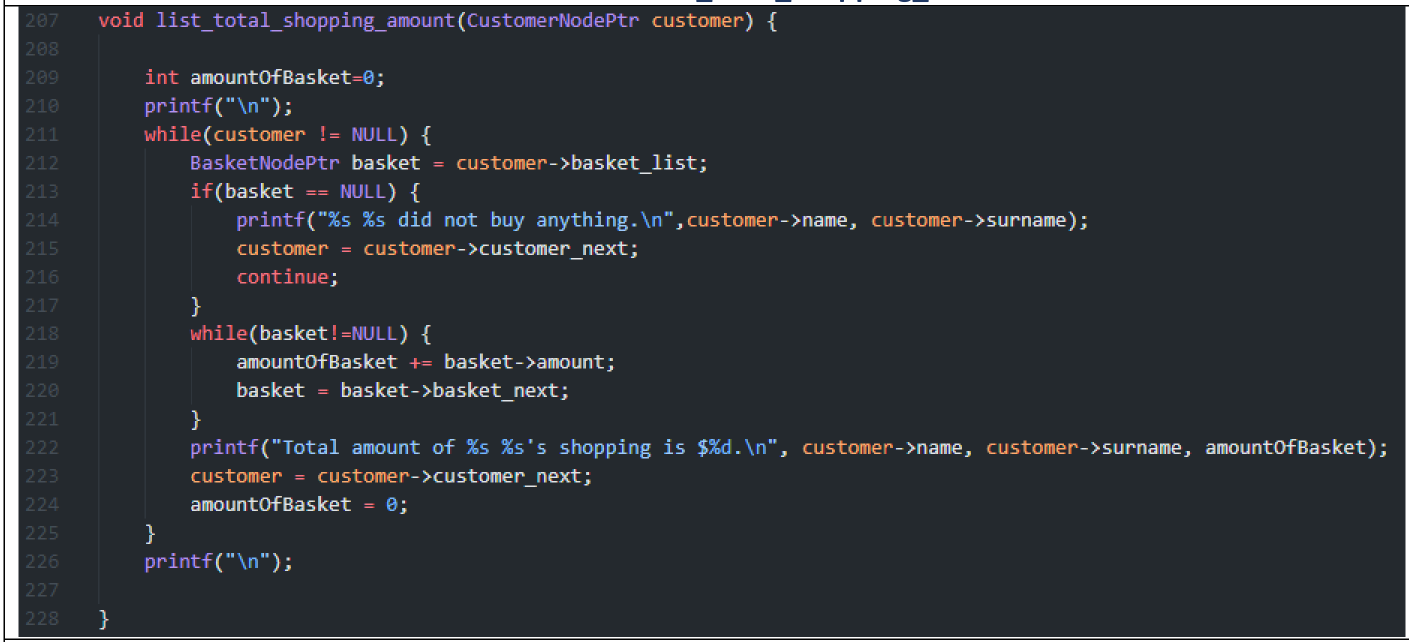
### PARAMETERS

- int custIDtoAddProduct: Takes customer's ID to add product to that customer's basket.
- int productIDToAdd: Takes product's ID to add that product to the related customer's basket.
- BasketNodePtr basketPtr: The related instance of struct basket.
- char menuChoice: Takes user's choice from the menu.
- ProductNodePtr *basketProductList: Takes the address of product list of related basket.
- ProductNodePtr currentProductPtr: The related instance of struct product.
- ProductNodePtr *product_list: Takes the address of an instance of struct product.
- CustomerNodePtr currentCustPtr: The related instance of struct customer.

### FUNCTION

- It inserts products according to basket.txt and the data entered by the user to the related basket. Also calculates the amount of that basket.
- If the user is adding a product as an input(this means if user choice is 2 from the menu) prints the products in the related basket with their properties and total amount of that basket.

## 7. FUNCTION list_total_shopping_amount



```c
void list_total_shopping_amount(CustomerNodePtr customer) {

    int amountOfBasket=0;
    printf("\n");
    while(customer != NULL) {
        BasketNodePtr basket = customer->basket_list;
        if(basket == NULL) {
            printf("%s %s did not buy anything.\n",customer->name, customer->surname);
            customer = customer->customer_next;
            continue;
        }
        while(basket!=NULL) {
            amountOfBasket += basket->amount;
            basket = basket->basket_next;
        }
        printf("Total amount of %s %s's shopping is $%d.\n", customer->name, customer->surname, amountOfBasket);
        customer = customer->customer_next;
        amountOfBasket = 0;
    }
    printf("\n");

}
```

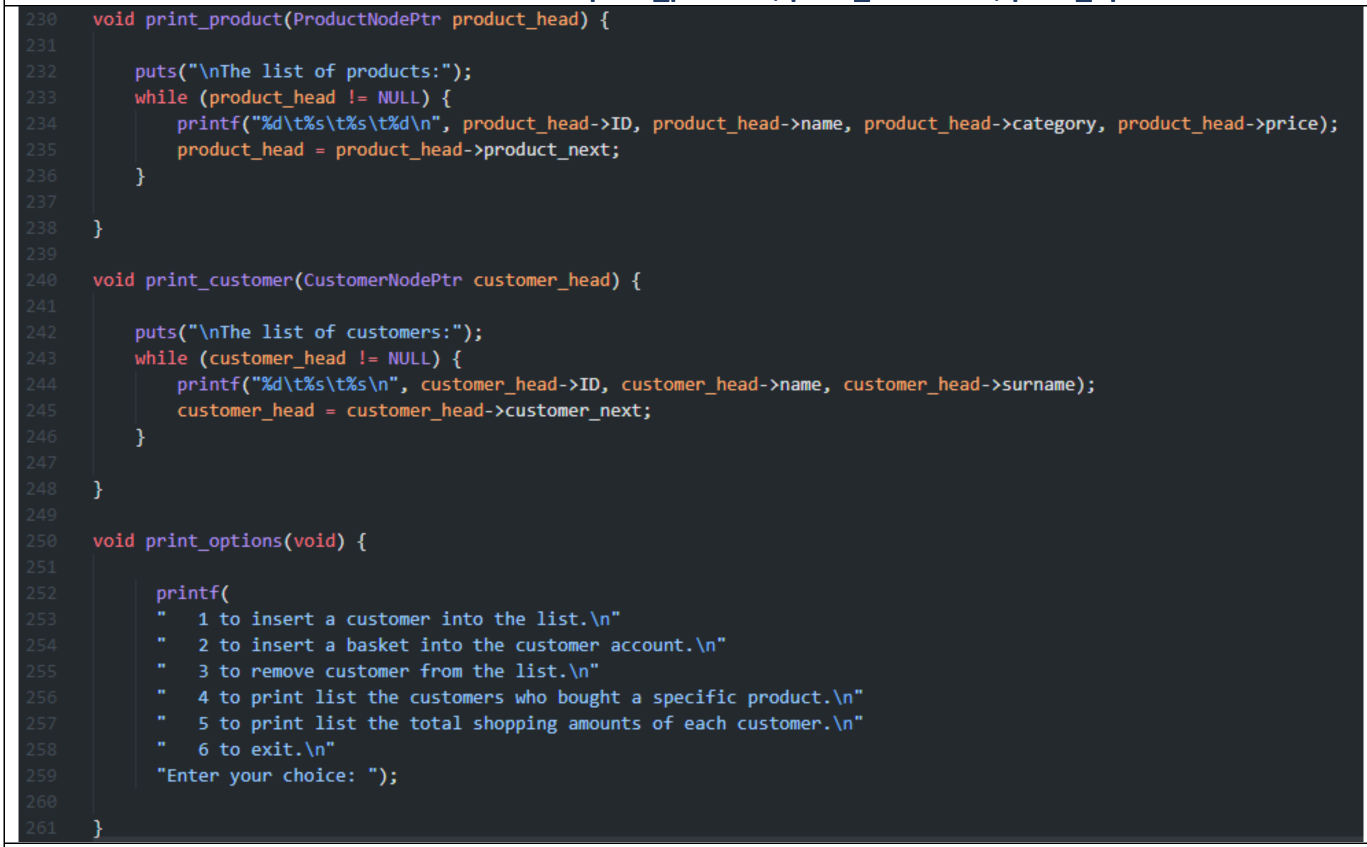
**PARAMETER**

- CustomerNodePtr customer: Takes the start pointer of customers.

**FUNCTION**

- It lists the total shopping amount of each customer. If a customer did not buy anything it prints their name with no amounts.

## 8. PRINT FUNCTIONS print_product, print_customer, print_options



```c
void print_product(ProductNodePtr product_head) {

    puts("\nThe list of products:");
    while (product_head != NULL) {
        printf("%d\t%s\t%s\t%d\n", product_head->ID, product_head->name, product_head->category, product_head->price);
        product_head = product_head->product_next;
    }

}

void print_customer(CustomerNodePtr customer_head) {

    puts("\nThe list of customers:");
    while (customer_head != NULL) {
        printf("%d\t%s\t%s\n", customer_head->ID, customer_head->name, customer_head->surname);
        customer_head = customer_head->customer_next;
    }

}

void print_options(void) {

      printf(
      "   1 to insert a customer into the list.\n"
      "   2 to insert a basket into the customer account.\n"
      "   3 to remove customer from the list.\n"
      "   4 to print list the customers who bought a specific product.\n"
      "   5 to print list the total shopping amounts of each customer.\n"
      "   6 to exit.\n"
      "Enter your choice: ");

}
```

**PARAMETERS**

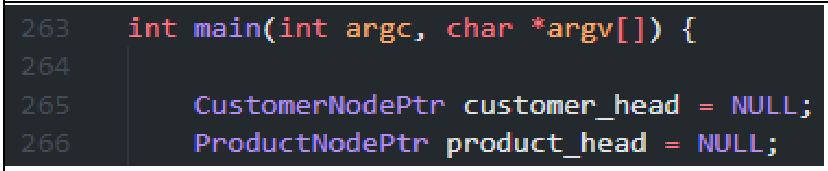
- ProductNodePtr product_head: Takes the start pointer of products as an address.
- CustomerNodePtr customer_head: Takes the start pointer of customers as an address.

**FUNCTIONS**

- First one prints products list.
- Second one prints customers list.
- Third one prints the menu.

**MAIN FUNCTION**

**GENERAL PARAMETERS**

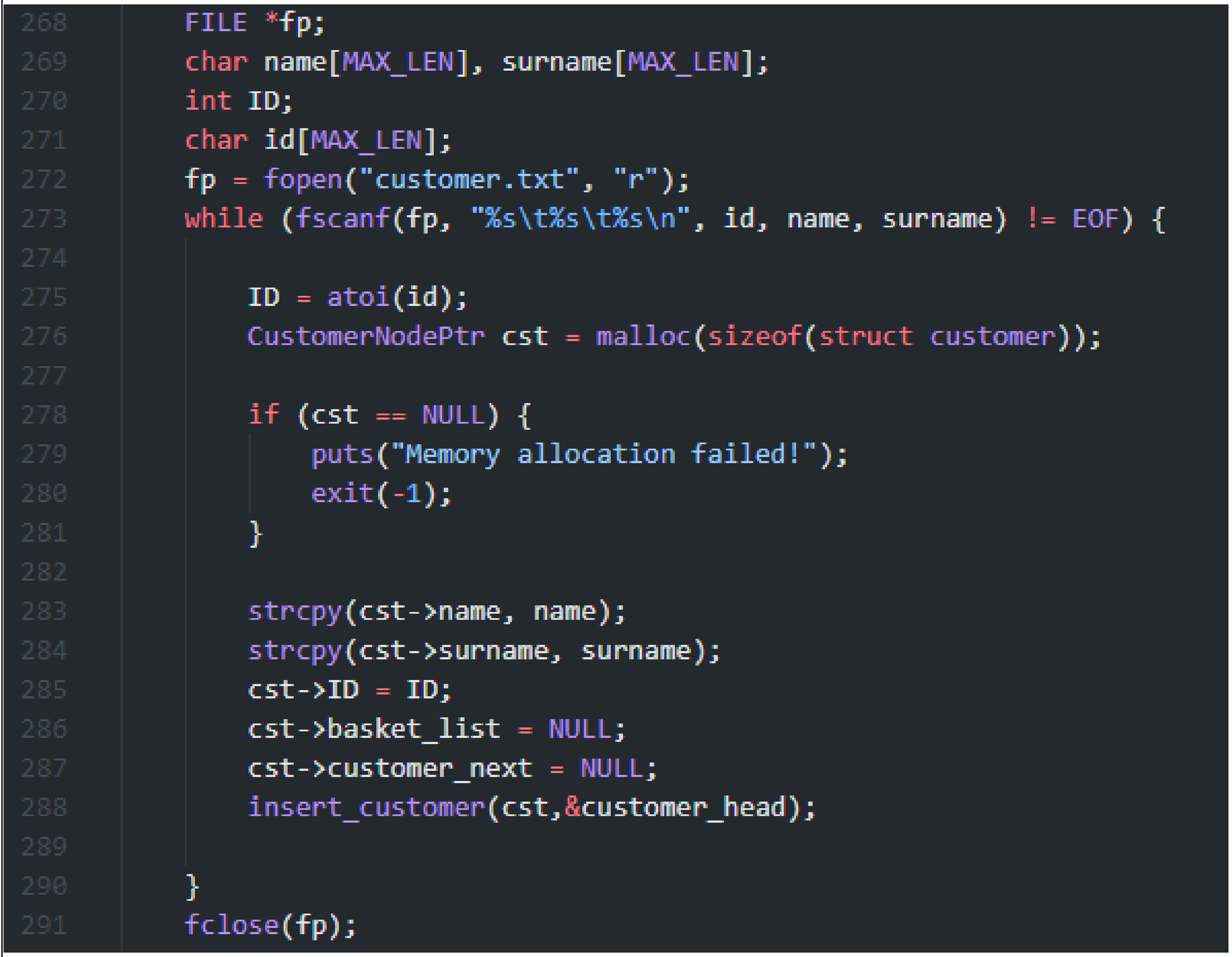


```c
int main(int argc, char *argv[]) {

    CustomerNodePtr customer_head = NULL;
    ProductNodePtr product_head = NULL;
```

- customer_head: Start pointer of customers linked list.
- product_head: Start pointer of products linked list.

**READ FILES**

**READ customer.txt**

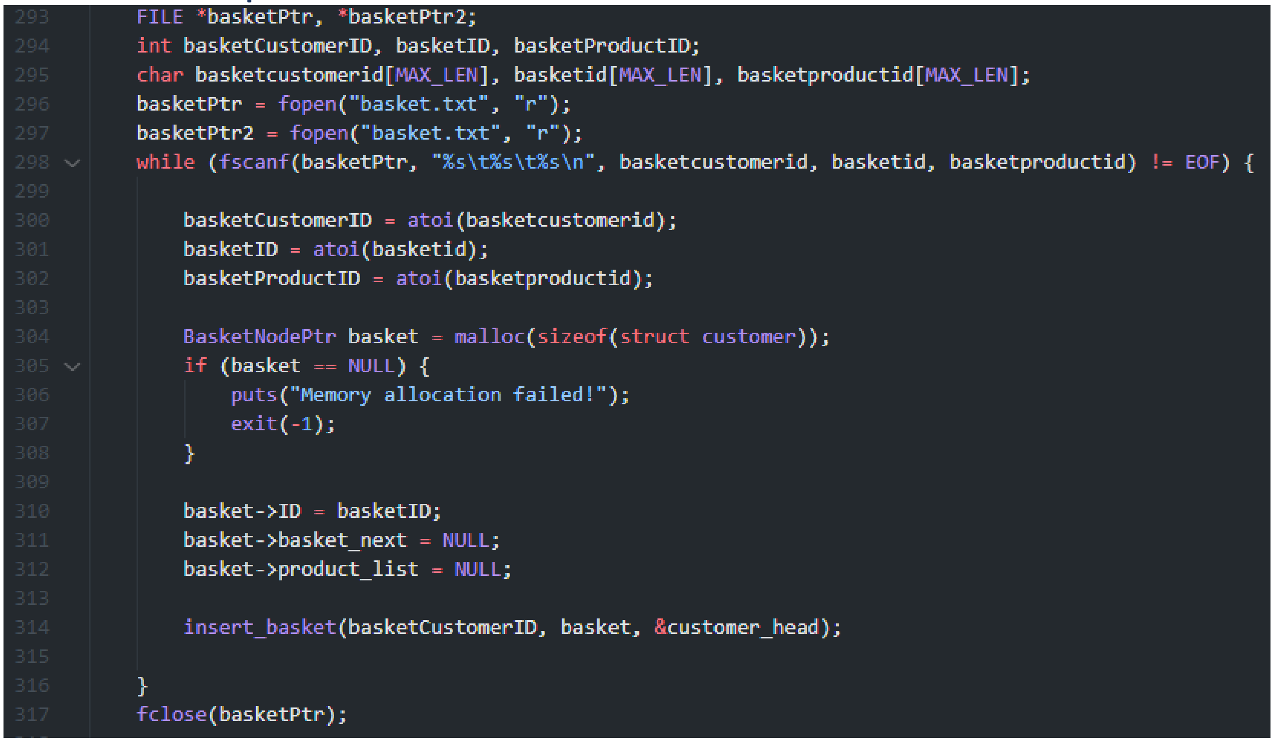


```c
    FILE *fp;
    char name[MAX_LEN], surname[MAX_LEN];
    int ID;
    char id[MAX_LEN];
    fp = fopen("customer.txt", "r");
    while (fscanf(fp, "%s\t%s\t%s\n", id, name, surname) != EOF) {

        ID = atoi(id);
        CustomerNodePtr cst = malloc(sizeof(struct customer));

        if (cst == NULL) {
            puts("Memory allocation failed!");
            exit(-1);
        }

        strcpy(cst->name, name);
        strcpy(cst->surname, surname);
        cst->ID = ID;
        cst->basket_list = NULL;
        cst->customer_next = NULL;
        insert_customer(cst,&customer_head);

    }
    fclose(fp);
```

Reads customer.txt and calls insert_customer function to add customers to the customer list.

**READ basket.txt**

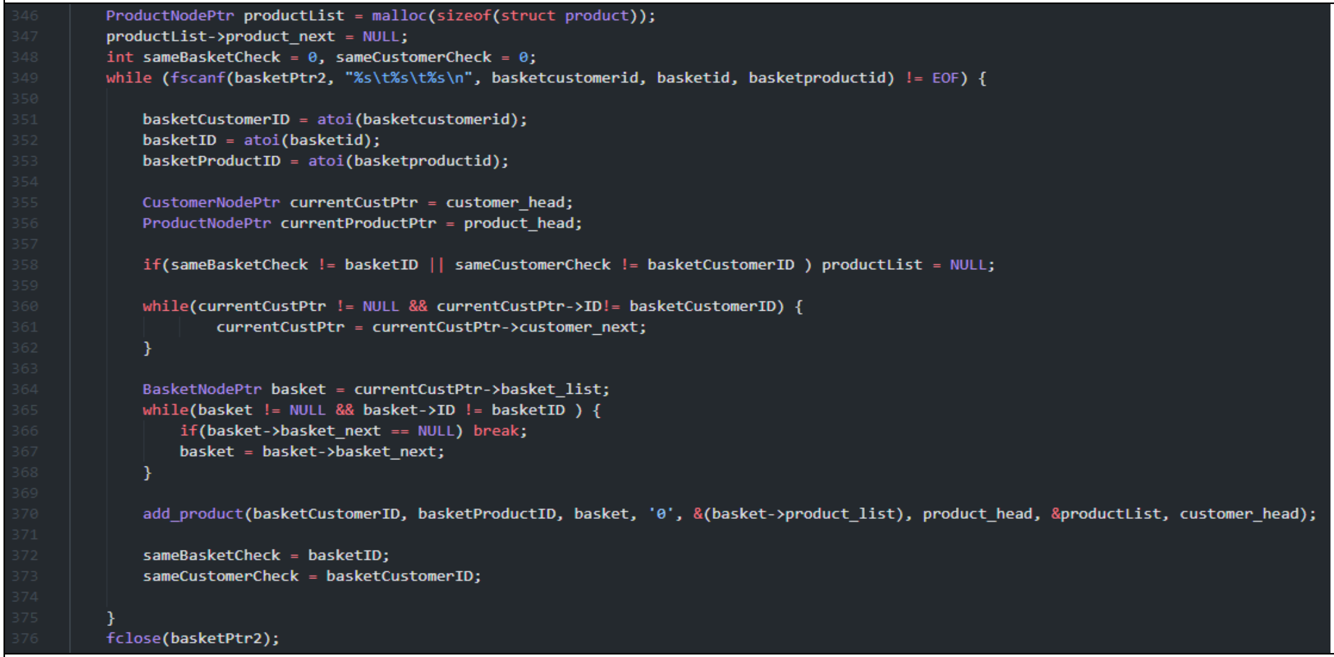
**BEFORE READING product.txt**



```c
    FILE *basketPtr, *basketPtr2;
    int basketCustomerID, basketID, basketProductID;
    char basketcustomerid[MAX_LEN], basketid[MAX_LEN], basketproductid[MAX_LEN];
    basketPtr = fopen("basket.txt", "r");
    basketPtr2 = fopen("basket.txt", "r");
    while (fscanf(basketPtr, "%s\t%s\t%s\n", basketcustomerid, basketid, basketproductid) != EOF) {

        basketCustomerID = atoi(basketcustomerid);
        basketID = atoi(basketid);
        basketProductID = atoi(basketproductid);

        BasketNodePtr basket = malloc(sizeof(struct customer));
        if (basket == NULL) {
            puts("Memory allocation failed!");
            exit(-1);
        }

        basket->ID = basketID;
        basket->basket_next = NULL;
        basket->product_list = NULL;

        insert_basket(basketCustomerID, basket, &customer_head);

    }
    fclose(basketPtr);
```

Reads basket.txt and calls insert_basket function to add baskets in addition to related customer’s baskets.

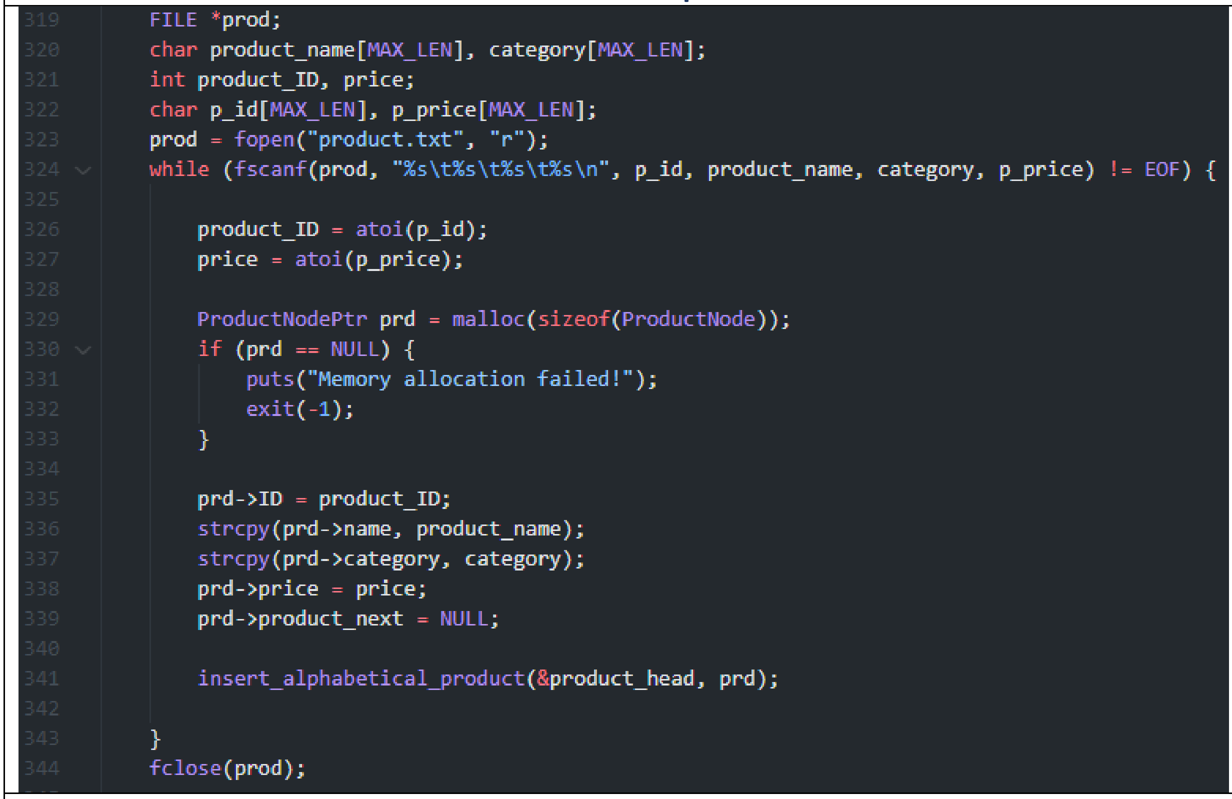
**AFTER READING product.txt**



```c
    ProductNodePtr productList = malloc(sizeof(struct product));
    productList->product_next = NULL;
    int sameBasketCheck = 0, sameCustomerCheck = 0;
    while (fscanf(basketPtr2, "%s\t%s\t%s\n", basketcustomerid, basketid, basketproductid) != EOF) {

        basketCustomerID = atoi(basketcustomerid);
        basketID = atoi(basketid);
        basketProductID = atoi(basketproductid);

        CustomerNodePtr currentCustPtr = customer_head;
        ProductNodePtr currentProductPtr = product_head;

        if(sameBasketCheck != basketID || sameCustomerCheck != basketCustomerID ) productList = NULL;

        while(currentCustPtr != NULL && currentCustPtr->ID!= basketCustomerID) {
                currentCustPtr = currentCustPtr->customer_next;
        }

        BasketNodePtr basket = currentCustPtr->basket_list;
        while(basket != NULL && basket->ID != basketID ) {
            if(basket->basket_next == NULL) break;
            basket = basket->basket_next;
        }

        add_product(basketCustomerID, basketProductID, basket, '0', &(basket->product_list), product_head, &productList, customer_head);

        sameBasketCheck = basketID;
        sameCustomerCheck = basketCustomerID;

    }
    fclose(basketPtr2);
```

Reads basket.txt and calls add_product function to add products to the related basket of the related customer.

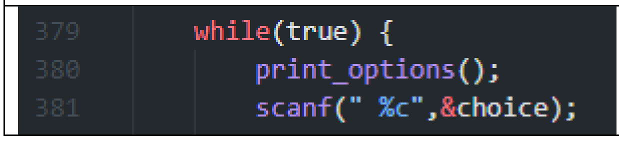
**READ product.txt**



```c
    FILE *prod;
    char product_name[MAX_LEN], category[MAX_LEN];
    int product_ID, price;
    char p_id[MAX_LEN], p_price[MAX_LEN];
    prod = fopen("product.txt", "r");
    while (fscanf(prod, "%s\t%s\t%s\t%s\n", p_id, product_name, category, p_price) != EOF) {

        product_ID = atoi(p_id);
        price = atoi(p_price);

        ProductNodePtr prd = malloc(sizeof(ProductNode));
        if (prd == NULL) {
            puts("Memory allocation failed!");
            exit(-1);
        }

        prd->ID = product_ID;
        strcpy(prd->name, product_name);
        strcpy(prd->category, category);
        prd->price = price;
        prd->product_next = NULL;

        insert_alphabetical_product(&product_head, prd);

    }
    fclose(prod);
```

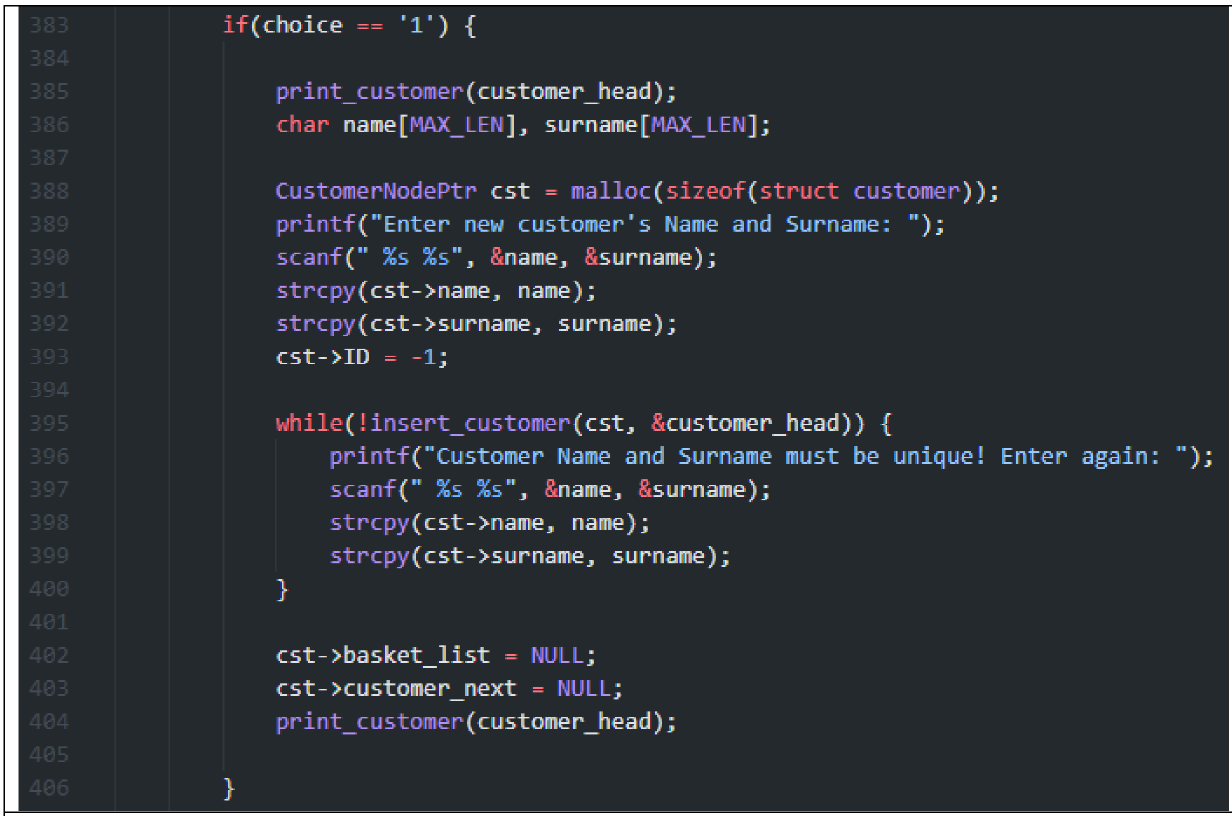
Reads product.txt and calls related function to sort products alphabetically.

**USER INPUT-OUTPUT**



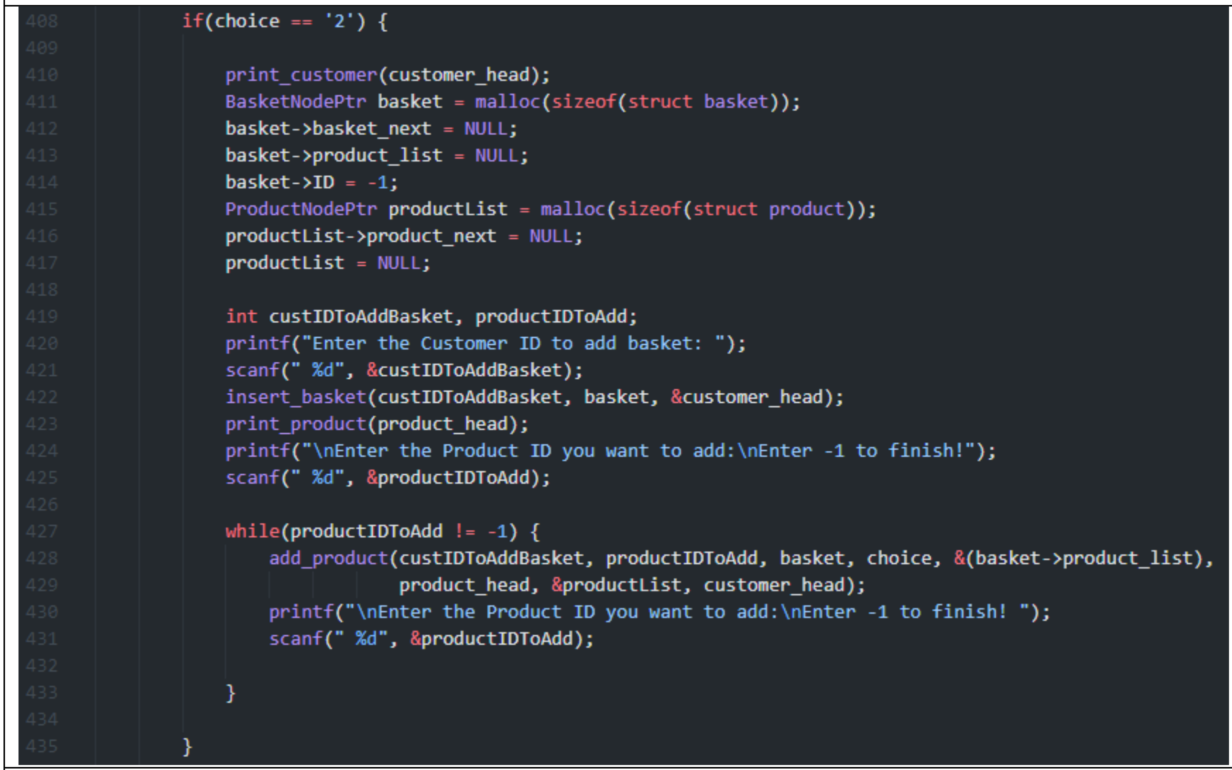
```c
    while(true) {
        print_options();
        scanf(" %c",&choice);
```

At first, calls prints_options to print the menu.



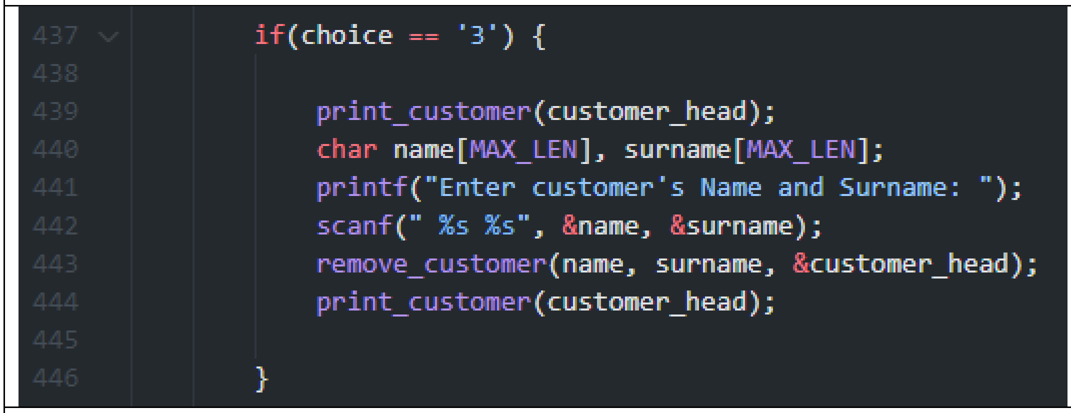
```c
        if(choice == '1') {

            print_customer(customer_head);
            char name[MAX_LEN], surname[MAX_LEN];

            CustomerNodePtr cst = malloc(sizeof(struct customer));
            printf("Enter new customer's Name and Surname: ");
            scanf(" %s %s", &name, &surname);
            strcpy(cst->name, name);
            strcpy(cst->surname, surname);
            cst->ID = -1;

            while(!insert_customer(cst, &customer_head)) {
                printf("Customer Name and Surname must be unique! Enter again: ");
                scanf(" %s %s", &name, &surname);
                strcpy(cst->name, name);
                strcpy(cst->surname, surname);
            }

            cst->basket_list = NULL;
            cst->customer_next = NULL;
            print_customer(customer_head);

        }
```

Asks customer information to the user for the customer which will be inserted to the customer list and if there is such a customer in the list already, gives a warning.



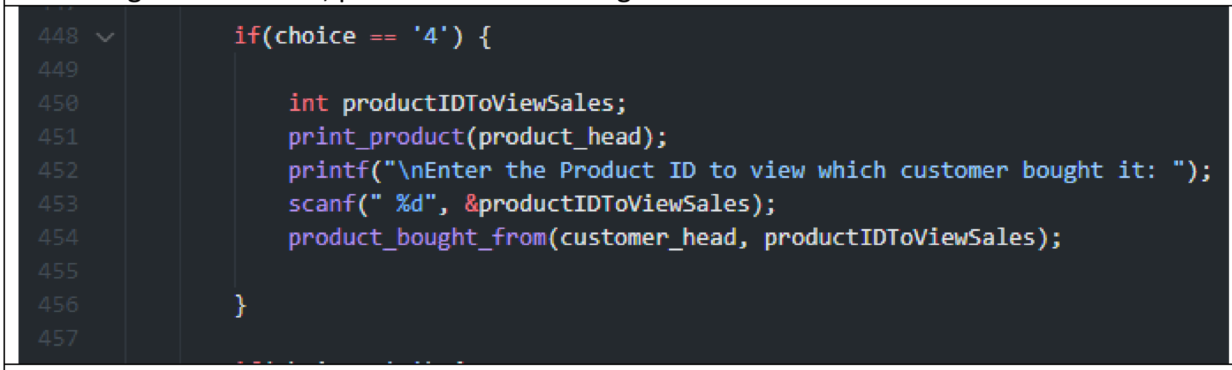
```c
        if(choice == '2') {

            print_customer(customer_head);
            BasketNodePtr basket = malloc(sizeof(struct basket));
            basket->basket_next = NULL;
            basket->product_list = NULL;
            basket->ID = -1;
            ProductNodePtr productList = malloc(sizeof(struct product));
            productList->product_next = NULL;
            productList = NULL;

            int custIDToAddBasket, productIDToAdd;
            printf("Enter the Customer ID to add basket: ");
            scanf(" %d", &custIDToAddBasket);
            insert_basket(custIDToAddBasket, basket, &customer_head);
            print_product(product_head);
            printf("\nEnter the Product ID you want to add:\nEnter -1 to finish!");
            scanf(" %d", &productIDToAdd);

            while(productIDToAdd != -1) {
                add_product(custIDToAddBasket, productIDToAdd, basket, choice, &(basket->product_list),
                            product_head, &productList, customer_head);
                printf("\nEnter the Product ID you want to add:\nEnter -1 to finish! ");
                scanf(" %d", &productIDToAdd);

            }

        }
```

Lists all customers, asks user to select a customer ID from customer list and product ID from product list. Until user enters -1 continues to add products and calculates the amount of related basket.



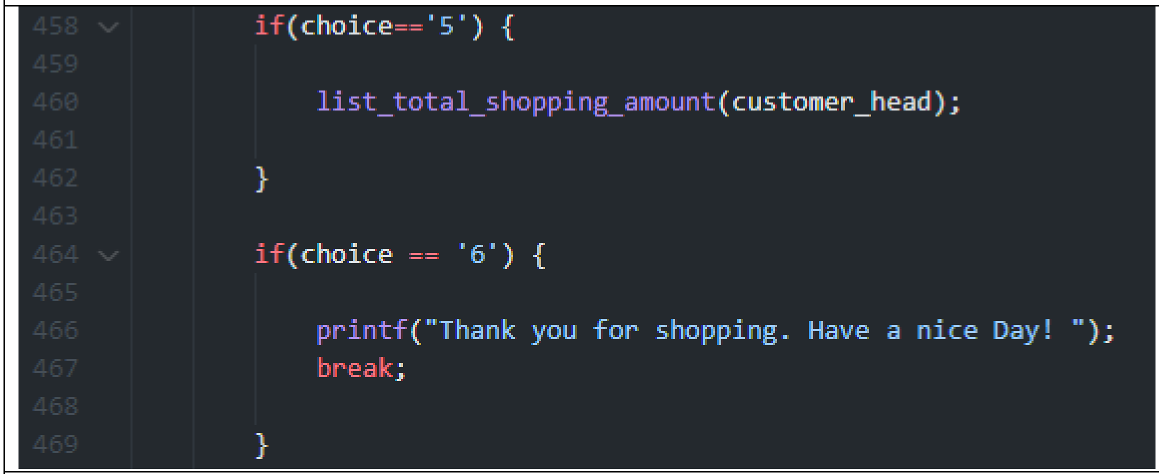
```c
if(choice == '3') {

    print_customer(customer_head);
    char name[MAX_LEN], surname[MAX_LEN];
    printf("Enter customer's Name and Surname: ");
    scanf(" %s %s", &name, &surname);
    remove_customer(name, surname, &customer_head);
    print_customer(customer_head);

}
```

Lists all customers. Asks user the customer information which is wanted to be removed and after removing the customer, prints all customers again.



```c
if(choice == '4') {

    int productIDToViewSales;
    print_product(product_head);
    printf("\nEnter the Product ID to view which customer bought it: ");
    scanf(" %d", &productIDToViewSales);
    product_bought_from(customer_head, productIDToViewSales);

}
```

Lists all products. Asks user the product ID and prints list of customers who bought the selected product.



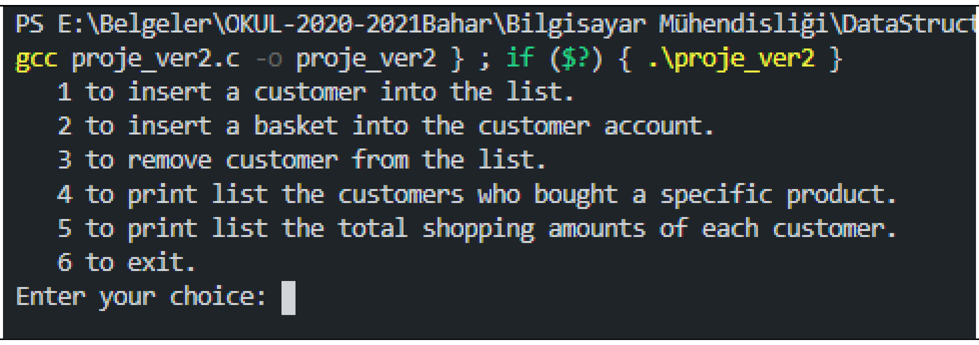
```c
if(choice=='5') {

    list_total_shopping_amount(customer_head);

}

if(choice == '6') {

    printf("Thank you for shopping. Have a nice Day! ");
    break;

}
```

If the user selects 5 from the menu, total shopping amount of each customer is listed. If some customers has no shopping, it prints their name with no amount.

If the user selects 6 from the menu, ends shopping process and exits.

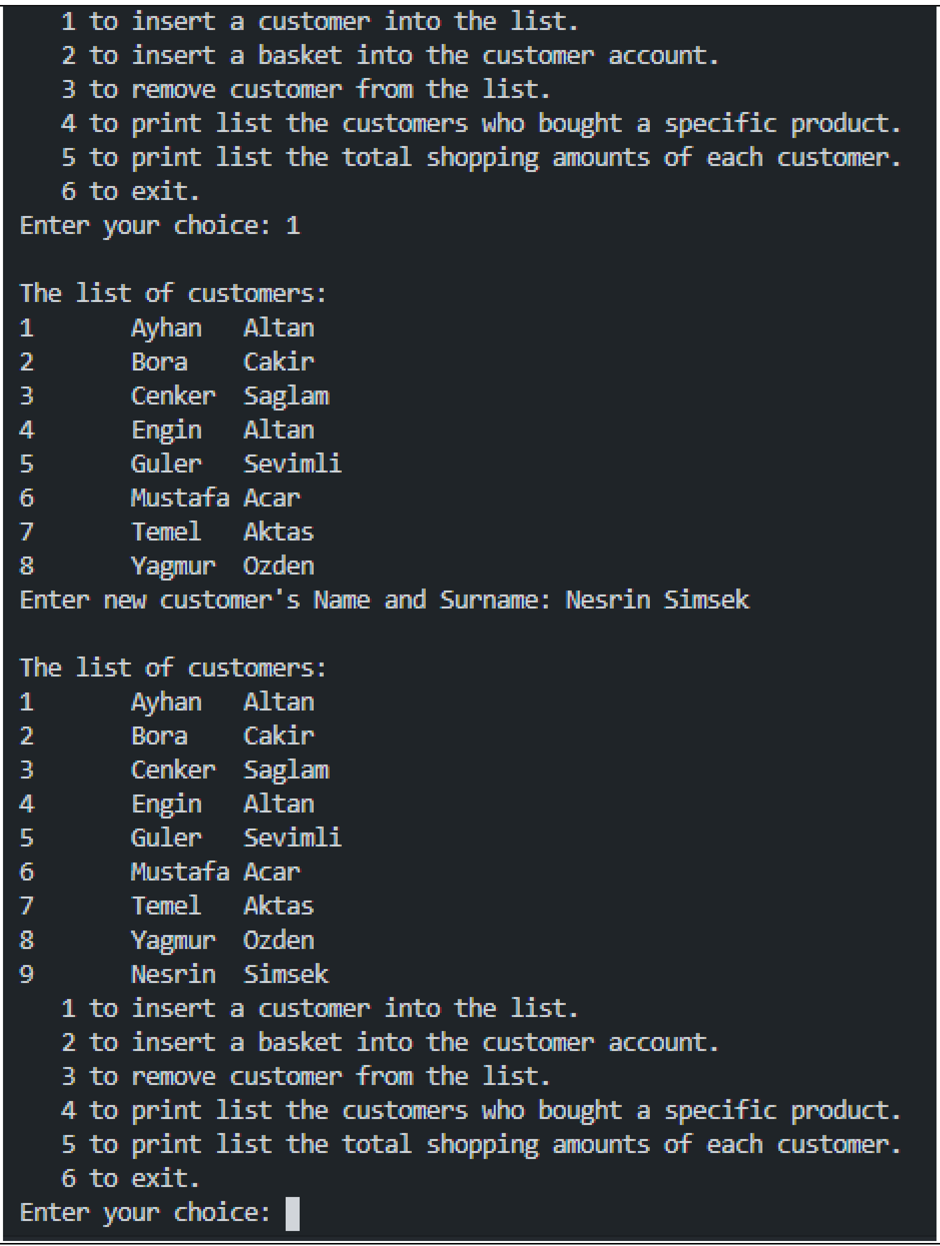
**EXECUTIONS**

**DEFAULT START**

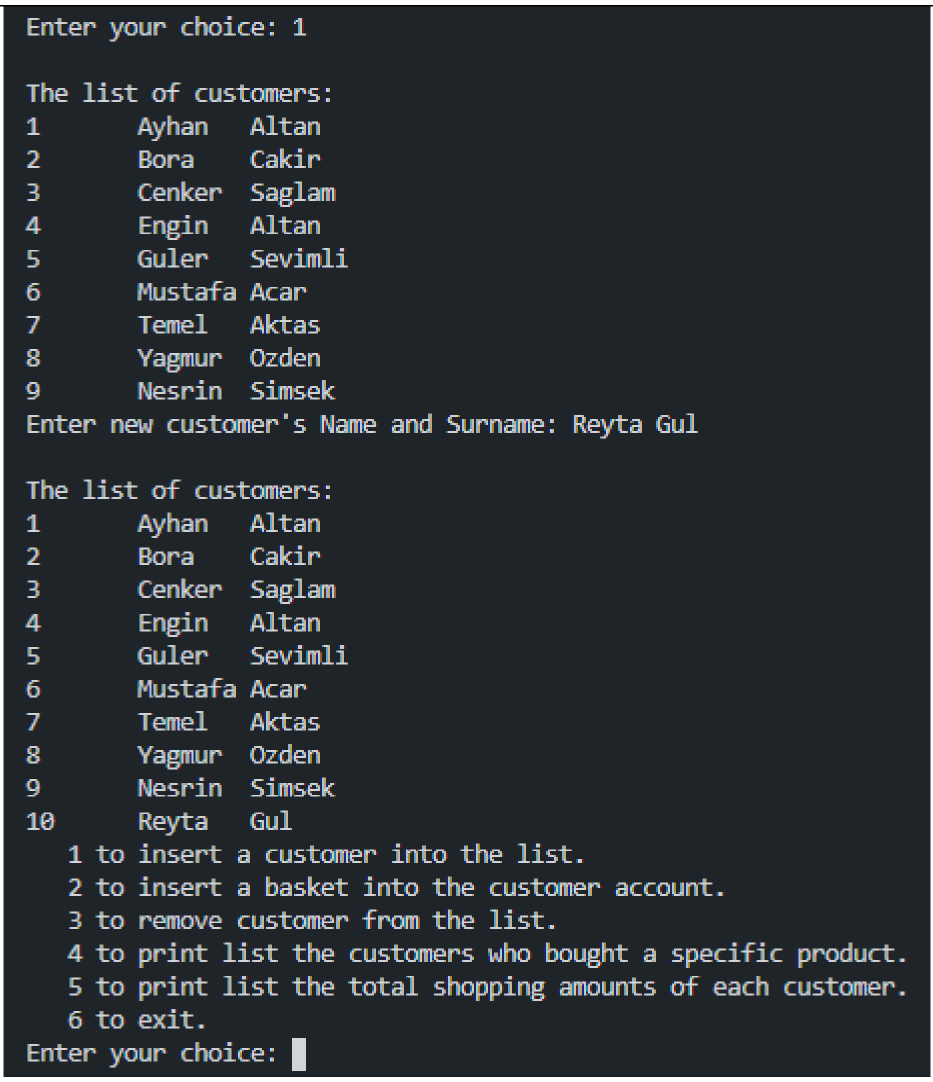


```
PS E:\Belgeler\OKUL-2020-2021Bahar\Bilgisayar Mühendisliği\DataStruct
gcc proje_ver2.c -o proje_ver2 } ; if ($?) { .\proje_ver2 }
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice:
```

**CHOICE 1**

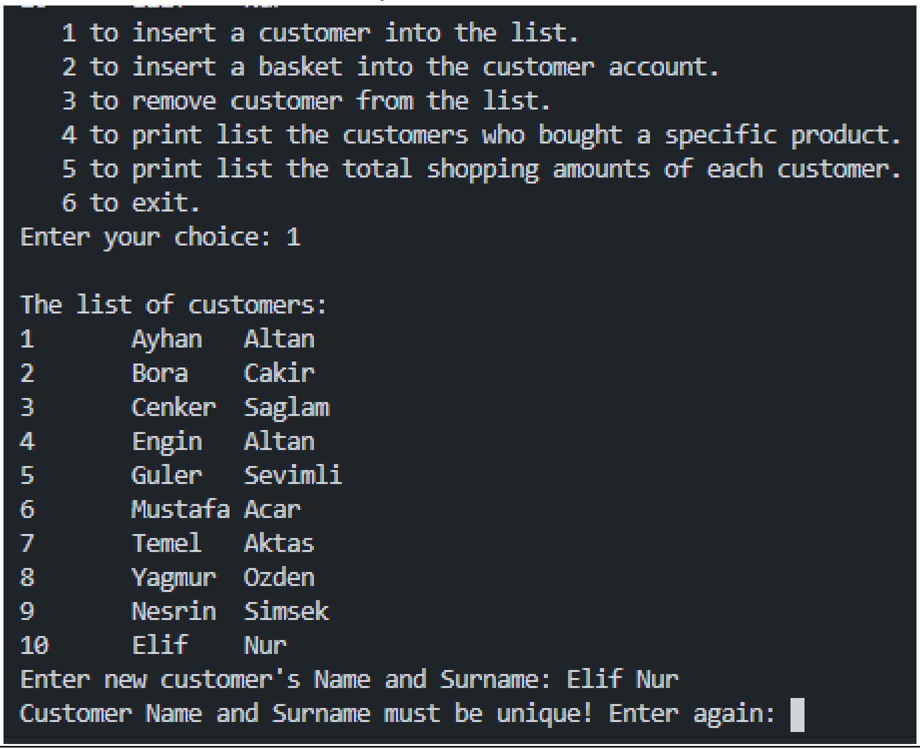


```
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice: 1

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
Enter new customer's Name and Surname: Nesrin Simsek

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice:
```



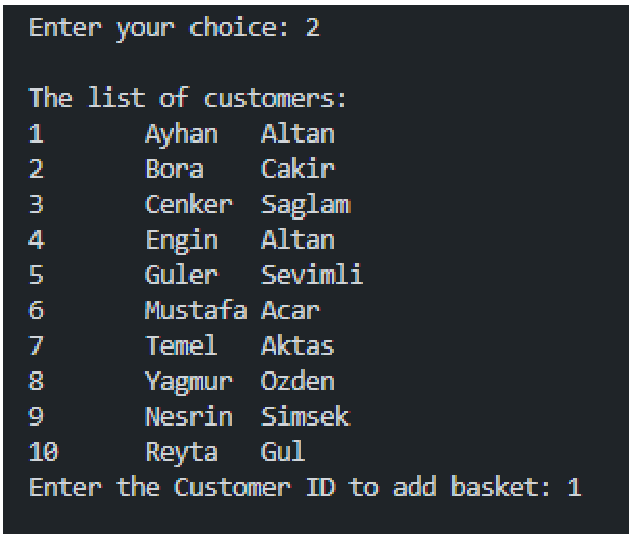
```
Enter your choice: 1

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
Enter new customer's Name and Surname: Reyta Gul

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
10      Reyta   Gul
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice: 
```

**UNIQUE NAME ERROR**



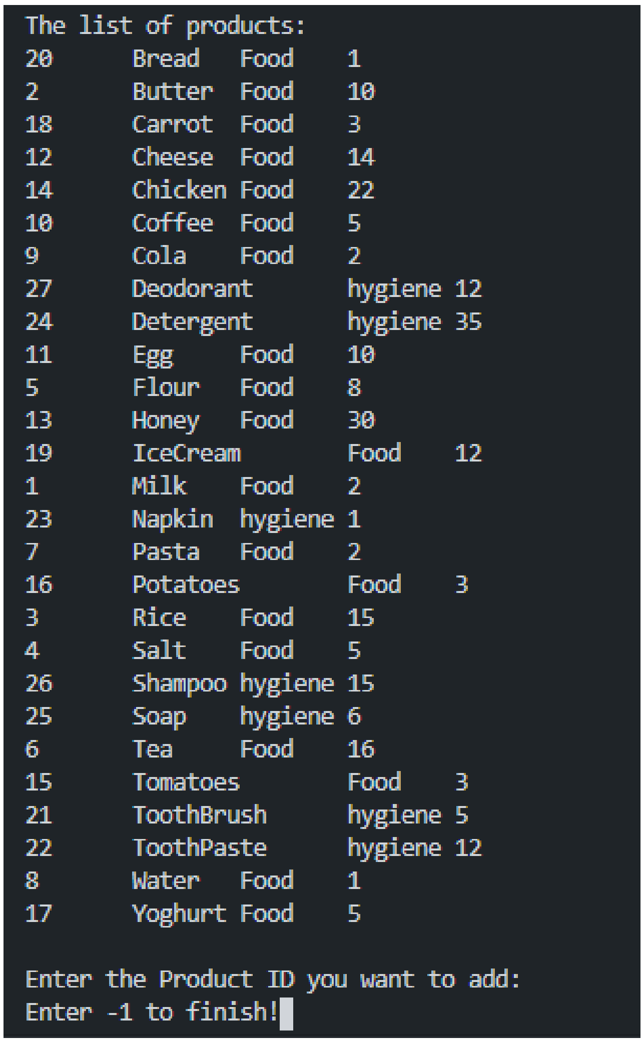
```
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice: 1

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
10      Elif    Nur
Enter new customer's Name and Surname: Elif Nur
Customer Name and Surname must be unique! Enter again: 
```

| CHOICE 2 |
|---|
| STEP 1 |



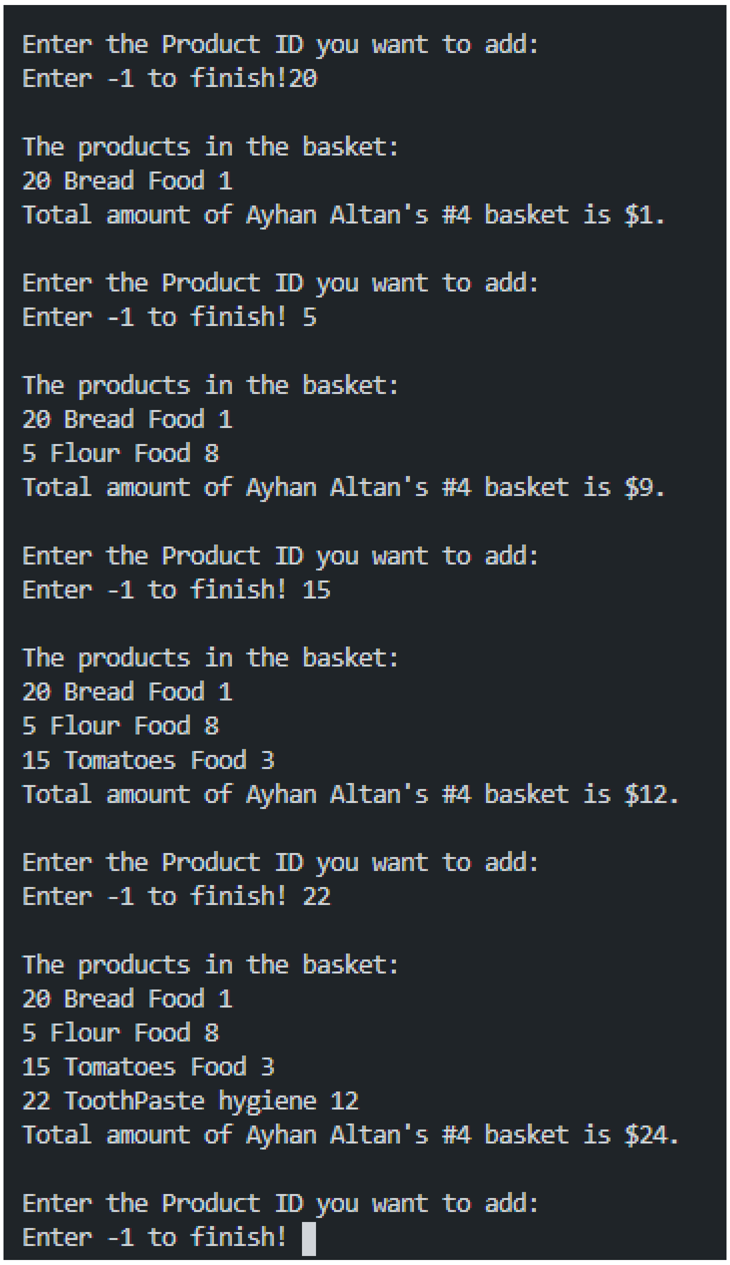
```
Enter your choice: 2

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
10      Reyta   Gul
Enter the Customer ID to add basket: 1
```

STEP 2



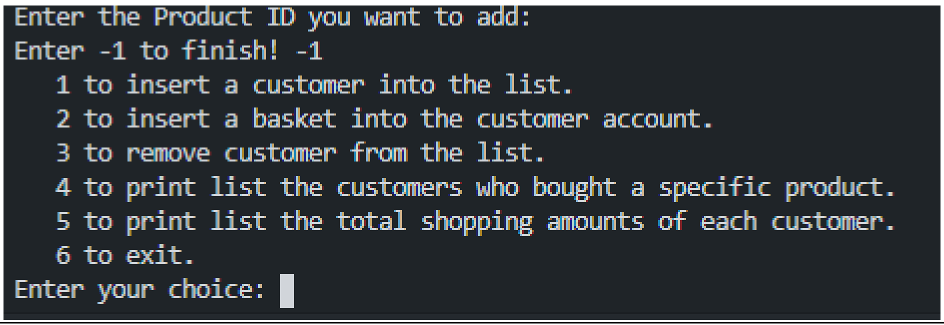
```
The list of products:
20      Bread   Food    1
2       Butter  Food    10
18      Carrot  Food    3
12      Cheese  Food    14
14      Chicken Food    22
10      Coffee  Food    5
9       Cola    Food    2
27      Deodorant       hygiene 12
24      Detergent       hygiene 35
11      Egg     Food    10
5       Flour   Food    8
13      Honey   Food    30
19      IceCream        Food    12
1       Milk    Food    2
23      Napkin  hygiene 1
7       Pasta   Food    2
16      Potatoes        Food    3
3       Rice    Food    15
4       Salt    Food    5
26      Shampoo hygiene 15
25      Soap    hygiene 6
6       Tea     Food    16
15      Tomatoes        Food    3
21      ToothBrush      hygiene 5
22      ToothPaste      hygiene 12
8       Water   Food    1
17      Yoghurt Food    5

Enter the Product ID you want to add:
Enter -1 to finish!
```

STEP 3



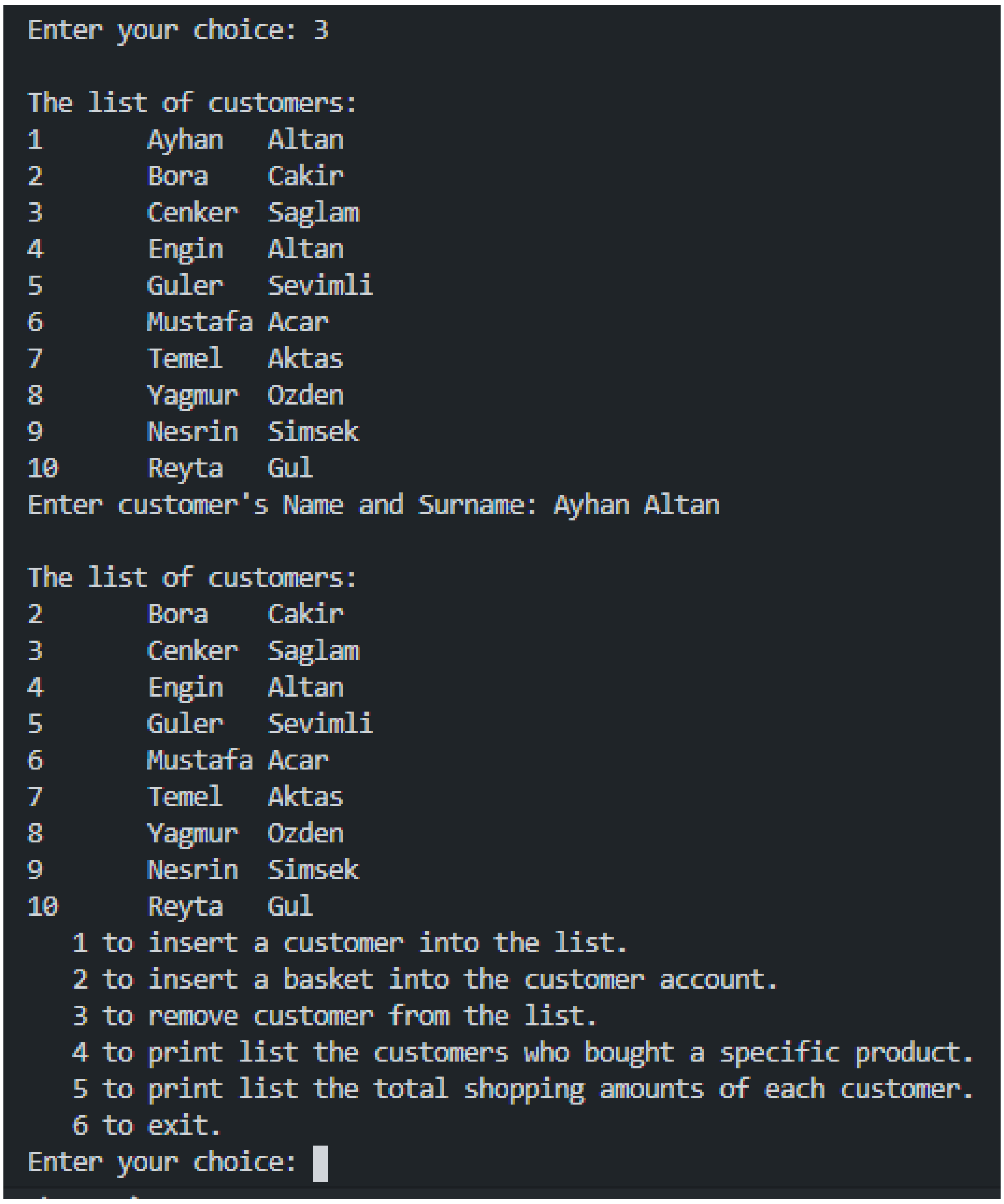
```
Enter the Product ID you want to add:
Enter -1 to finish!20

The products in the basket:
20 Bread Food 1
Total amount of Ayhan Altan's #4 basket is $1.

Enter the Product ID you want to add:
Enter -1 to finish! 5

The products in the basket:
20 Bread Food 1
5 Flour Food 8
Total amount of Ayhan Altan's #4 basket is $9.

Enter the Product ID you want to add:
Enter -1 to finish! 15

The products in the basket:
20 Bread Food 1
5 Flour Food 8
15 Tomatoes Food 3
Total amount of Ayhan Altan's #4 basket is $12.

Enter the Product ID you want to add:
Enter -1 to finish! 22

The products in the basket:
20 Bread Food 1
5 Flour Food 8
15 Tomatoes Food 3
22 ToothPaste hygiene 12
Total amount of Ayhan Altan's #4 basket is $24.

Enter the Product ID you want to add:
Enter -1 to finish!
```

EXIT FROM ADDING PRODUCT



```
Enter the Product ID you want to add:
Enter -1 to finish! -1
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice:
```

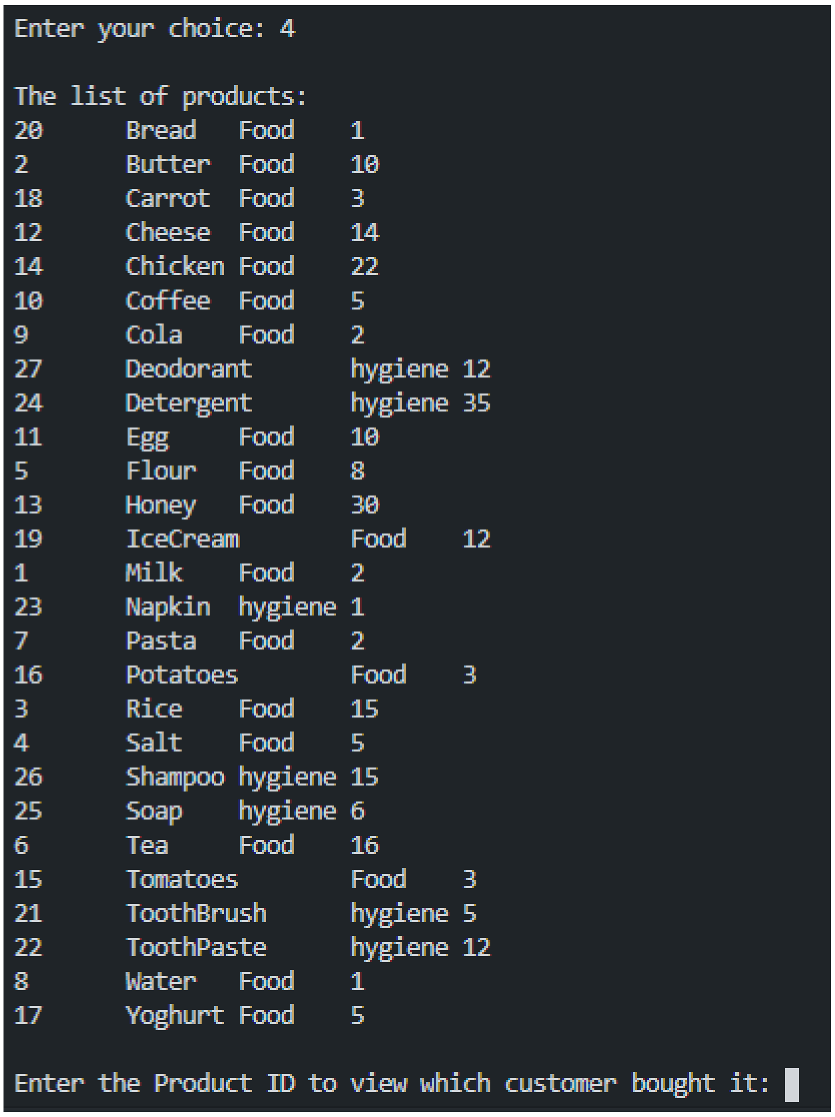
**CHOICE 3**



```
Enter your choice: 3

The list of customers:
1       Ayhan   Altan
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
10      Reyta   Gul
Enter customer's Name and Surname: Ayhan Altan

The list of customers:
2       Bora    Cakir
3       Cenker  Saglam
4       Engin   Altan
5       Guler   Sevimli
6       Mustafa Acar
7       Temel   Aktas
8       Yagmur  Ozden
9       Nesrin  Simsek
10      Reyta   Gul
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice: 
```

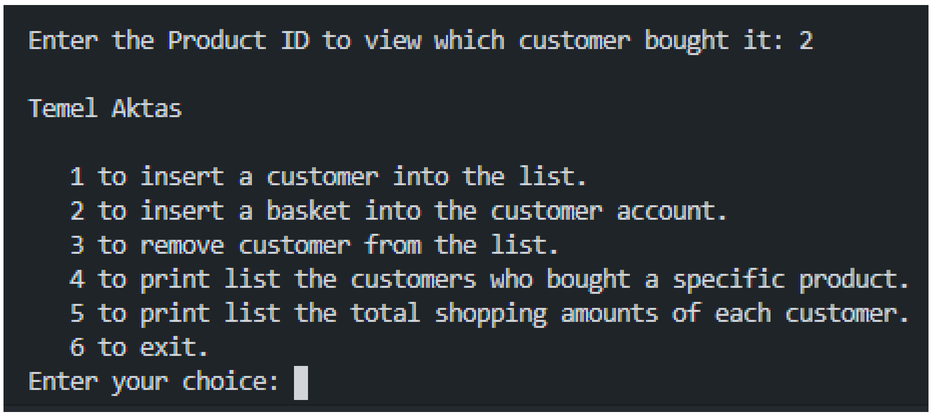
**CHOICE 4**

STEP 1

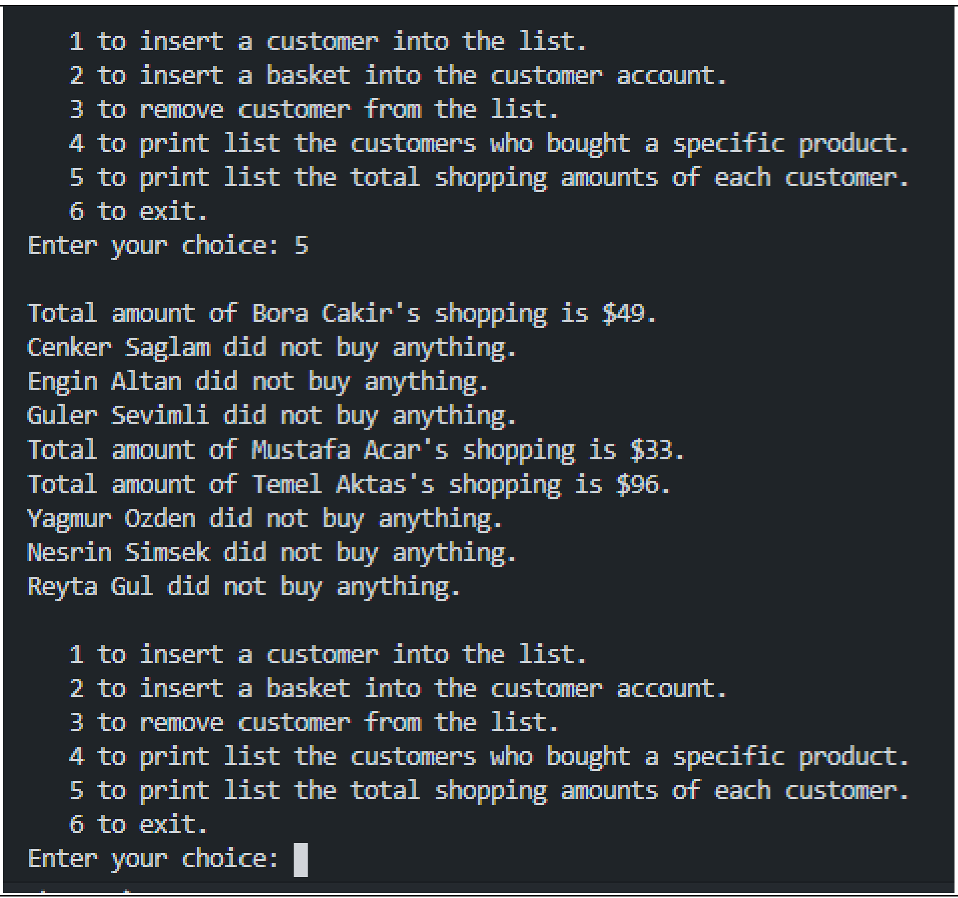
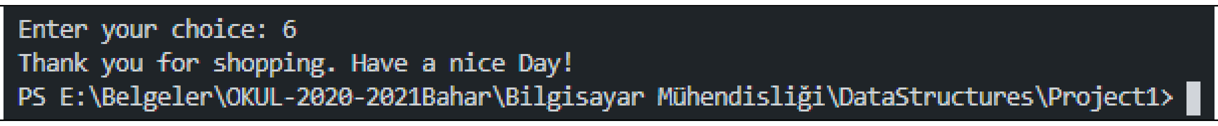


```
Enter your choice: 4

The list of products:
20      Bread   Food    1
2       Butter  Food    10
18      Carrot  Food    3
12      Cheese  Food    14
14      Chicken Food    22
10      Coffee  Food    5
9       Cola    Food    2
27      Deodorant       hygiene 12
24      Detergent       hygiene 35
11      Egg     Food    10
5       Flour   Food    8
13      Honey   Food    30
19      IceCream        Food    12
1       Milk    Food    2
23      Napkin  hygiene 1
7       Pasta   Food    2
16      Potatoes        Food    3
3       Rice    Food    15
4       Salt    Food    5
26      Shampoo hygiene 15
25      Soap    hygiene 6
6       Tea     Food    16
15      Tomatoes        Food    3
21      ToothBrush      hygiene 5
22      ToothPaste      hygiene 12
8       Water   Food    1
17      Yoghurt Food    5

Enter the Product ID to view which customer bought it:
```

STEP 2



```
Enter the Product ID to view which customer bought it: 2

Temel Aktas

   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice:
```

| CHOICE 5 |
|---|
|  |
| CHOICE 6 |
|  |

```
   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice: 5

Total amount of Bora Cakir's shopping is $49.
Cenker Saglam did not buy anything.
Engin Altan did not buy anything.
Guler Sevimli did not buy anything.
Total amount of Mustafa Acar's shopping is $33.
Total amount of Temel Aktas's shopping is $96.
Yagmur Ozden did not buy anything.
Nesrin Simsek did not buy anything.
Reyta Gul did not buy anything.

   1 to insert a customer into the list.
   2 to insert a basket into the customer account.
   3 to remove customer from the list.
   4 to print list the customers who bought a specific product.
   5 to print list the total shopping amounts of each customer.
   6 to exit.
Enter your choice:
```

```
Enter your choice: 6
Thank you for shopping. Have a nice Day!
PS E:\Belgeler\OKUL-2020-2021Bahar\Bilgisayar Mühendisliği\DataStructures\Project1>
```