INF211
ALGORITHMS AND PROGRAMMING

PROJECT-12

Create your own Project

Deadline is Monday, January 11, 2021 at 17:00.

Projects that are not delivered on time are not accepted.

Upload the project to the Final Project 12 assignment section of the INF211 class.

The questions can be asked to course lecturer Dr. Tuba GÖZEL and teaching assistant written in the project list excel file under the Project 12 announcement.

Complete your project, fill the report by using attached word file (INF211_Project_12_Report_Template.docx).

Each team member should upload the project codes and the report.
You can compress the project codes if number of the files is more than ten.

The file name of report should be as *Project12_ProgramName_ProjectName_.docx* (i.e. *P12_Quiz_Show_Proje_Yılmaz_Kaya.docx*).

INF 211 Algorithms and Programming

2020-2021 Fall
Electronic Engineering

Project ID: 53_59_98

Project Name
Proje_Aygün_Tumbul_Eren

Program Name
SEK-TEK

Team Members

1901022022	**Selimhan**	**Aygün**
1901022030	**Eren**	**Tumbul**
19010220	**Ümit Kerem**	**Teke**

PROJECT OBJECTIVE

Briefly describe the objectives of your project

To write a practical user-based electronic kiosk program with more than one restaurant, special wallet and currency.

PROBLEM

Briefly describe the problem you given to solve

With the effect of modern life style and shopping mall culture, every area where people live has become crowded. As a result, just ordering at the points where fast-food chains with thousands of visitors a day are clustered has turned into a torture. In this age where speed is at the forefront, even fast food chains are not fast enough and cannot meet the needs.

ANALYSIS

Analyze of the problem

The problem is entirely due to the lack of speed. Kiosks with written software can be placed in various places and eliminate the queuing system.
A kiosk software can be created that saves people from queuing in long queues, collects restaurants under one roof and takes orders. By providing users with a fast and simple interface, orders can be taken quickly. Orders can be received much faster with the wallet system. Two different payment methods can be offered to diversify the payment. Orders received are instantly sent to the local terminals of the restaurants and a message can be sent when the order is ready.

DATA REQUIREMENTS

Specify all data requirements of the problem

Structured Data Type

```
struct
{
   char name[20];
   char surname[20];
   char username[20];
   char password[20];
   char mail[50];
   unsigned long long int pnumber;
   float deposit;
   float load;
}l,check,upd,lgn;
//cart structs
struct Cart{
        char product[30];
        float product_price;
        int product_count;
        char drink_choice[30];
        char drink_type[30];
        int drink_count;
        float drink_price;
        char pizza_choice[30];
        int pizza_choice_count;
};
        struct Cart cart[5] ={
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0}
        };
//data structs
struct menu{
        char menu[30];
        float menu_price;
        char extra1[30];
        char extra2[30];
        char extra3[30];
        char extra4[30];
};
struct pizza{
        char small[30];
        float small_price;
        char middle[30];
        float middle_price;
        char big[30];
        float big_price;
```

```
};
struct addon{
	char drink[30]; //cola
	float drink_price; //fiyat
	char addons[30];  //patates
	char addon_char[30]; //litre 2.5litre buyuk orta kucuk
	float addon_price_gap;
};

	struct sb_data_som{ //single or menu
		struct menu *menu;
		struct pizza *pizza;
		struct addon *addon;
	};
		struct menu menu[3]= {
		{"taraftar menu", 29.95, "kucuk boy pizza","tereyagli firin patates","kutu
icecek", "bir kisilik menu" },
		{"algida menusu", 34.99, "orta boy pizza","iki adet nogger dondurma","bir
litre icecek" },
		{"muhtesem uclu menu",34.99,"orta boy pizza","tereyagli firin patates","bir
litre icecek"}
		};
		struct pizza pizza[5]= {
			{"sucuklu kucuk pizza", 21.5, "sucuklu orta boy pizza", 34.5,
"sucuklu buyuk boy pizza",41.5},
			{"kucuk boy surpreme pizza", 27.5,"orta boy surpreme pizza",
34.5,"buyuk boy surpreme pizza",41.5},
			{"kucuk boy milano pizza", 27.5,"orta boy milano
pizza",34.5,"buyuk boy milano pizza,",41.5},
			{"kucuk boy gennaro pizza", 27.5,"orta boy gennaro
pizza",34.5,"buyuk boy gennaro pizza,",41.5},
			{"kucuk boy venedikli pizza", 24.5,"orta boy venedikli
pizza",31.5,"buyuk boy gennaro pizza,",38.5}
		};
		struct addon addon[5]= {
			{"coco cola ",4.0,"baharat","kucuk boy",0.0},
			{"fanta",3.5,"firin patates","orta boy",5.50},
			{"sprite", 8.0,"zeytin","buyuk boy",0.0},
			{"nuko cola", 12.0,"kasar peyniri kenarlikli","2.5 litre", 5.50},
			{"fuse tea",2.5,"kekik","1 litre",0.0}
		};
```

Problem Constants

We did not use any constant.

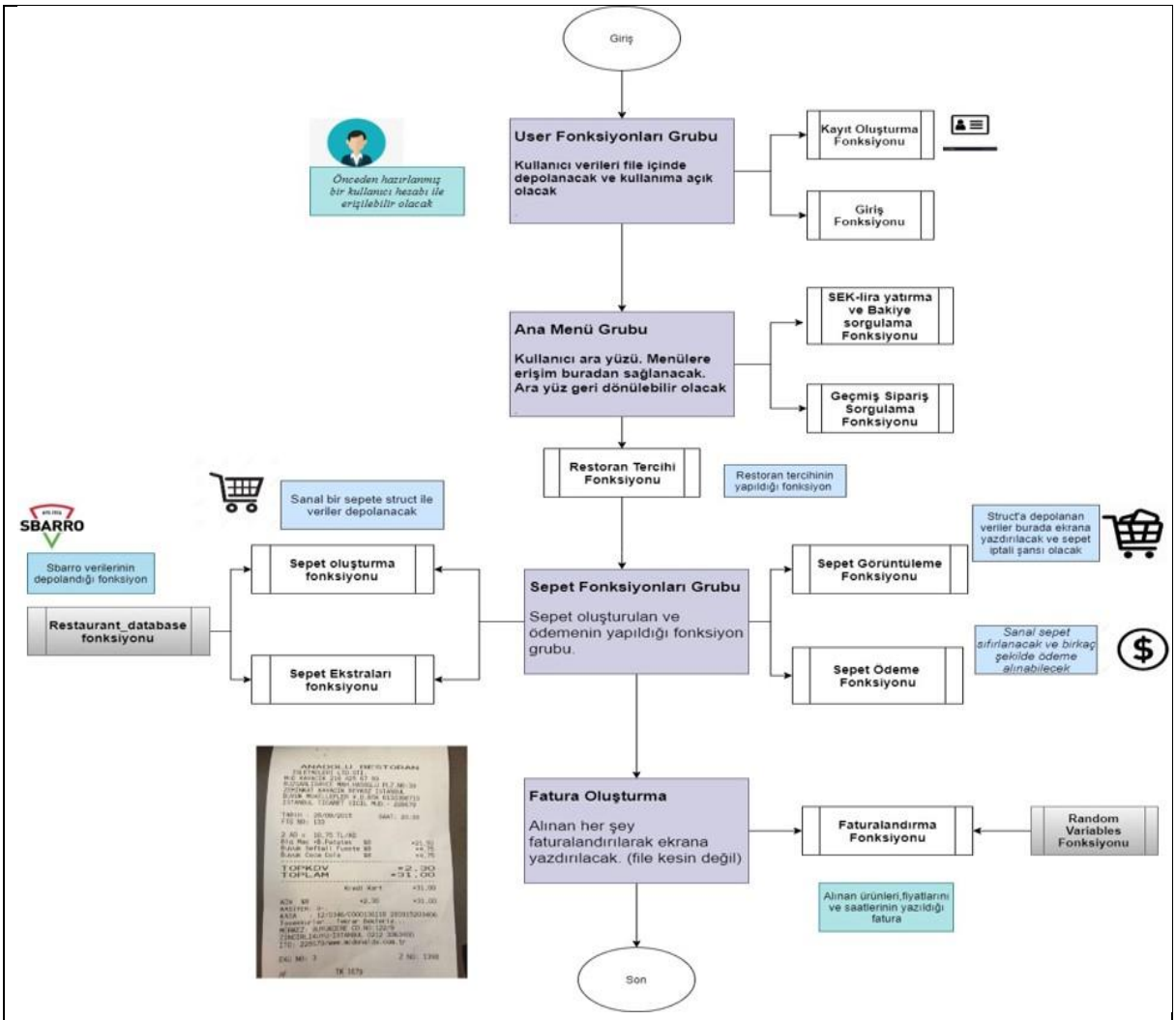
Problem Inputs

int option //in welcome menu to choose option
struct char mail[50]
struct char l.username[20]
struct char l.password[20]
struct char l.name[20]
struct char l.surname[20]
struct unsignef long long int l.pnumber
int main_exit
int procedure
float pr
int c
int quantity
struct float upd.load
int choose
struct char upd.mail[50]
struct char upd.password[20]
struct char upd.name[20]
struct char upd.surname[20]
struct long long int upd.pnumber
int temp
char r_name[30]
int p
int ch
int count
int p_choice
int menu_counter
int brak
int r
int counter=0
int p_count
int counter=0
int dtc,dc,quantity
int pc
int pq
int i
float total
int chc, cash
char userfile[20];

Problem Output

int cart_displayer
float change;

DESIGN
ALGORITHM
Design algorithm of the problem



HEADER F;ILE

```
#ifndef SEKTEK_H_
#define SEKTEK_H_
struct
{
   char name[20];
   char surname[20];
   char username[20];
   char password[20];
   char mail[50];
```

```
    unsigned long long int pnumber;
    float deposit;
    float load;
}l,check,upd,lgn;
//cart structs
struct Cart{
        char product[30];
        float product_price;
        int product_count;
        char drink_choice[30];
        char drink_type[30];
        int drink_count;
        float drink_price;
        char pizza_choice[30];
        int pizza_choice_count;
};
        struct Cart cart[5] ={
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0},
                {"empty",0.0,0,"empty","empty",0,0.0,"empty",0}
        };

//data structs
struct menu{
        char menu[30];
        float menu_price;
        char extra1[30];
        char extra2[30];
        char extra3[30];
        char extra4[30];
};
struct pizza{
        char small[30];
        float small_price;
        char middle[30];
        float middle_price;
        char big[30];
        float big_price;
};
struct addon{
        char drink[30]; //cola
        float drink_price; //fiyat
        char addons[30];  //patates
        char addon_char[30]; //litre 2.5litre buyuk orta kucuk
        float addon_price_gap;
};
struct sb_data_som{ //single or menu
                struct menu *menu;
```

```
                struct pizza *pizza;
                struct addon *addon;
        };
                struct menu menu[3]= {
                {"taraftar menu", 29.95, "kucuk boy pizza","tereyagli firin patates","kutu
icecek", "bir kisilik menu" },
                {"algida menusu", 34.99, "orta boy pizza","iki adet nogger dondurma","bir
litre icecek" },
                {"muhtesem uclu menu",34.99,"orta boy pizza","tereyagli firin patates","bir
litre icecek"}
                };
                struct pizza pizza[5]= {
                        {"sucuklu kucuk pizza", 21.5, "sucuklu orta boy pizza", 34.5,
"sucuklu buyuk boy pizza",41.5},
                        {"kucuk boy surpreme pizza", 27.5,"orta boy surpreme pizza",
34.5,"buyuk boy surpreme pizza",41.5},
                        {"kucuk boy milano pizza", 27.5,"orta boy milano
pizza",34.5,"buyuk boy milano pizza,",41.5},
                        {"kucuk boy gennaro pizza", 27.5,"orta boy gennaro
pizza",34.5,"buyuk boy gennaro pizza,",41.5},
                        {"kucuk boy venedikli pizza", 24.5,"orta boy venedikli
pizza",31.5,"buyuk boy gennaro pizza,",38.5}
                };
                struct addon addon[5]= {
                        {"coco cola ",4.0,"baharat","kucuk boy",0.0},
                        {"fanta",3.5,"firin patates","orta boy",5.50},
                        {"sprite", 8.0,"zeytin","buyuk boy",0.0},
                        {"nuko cola", 12.0,"kasar peyniri kenarlikli","2.5 litre", 5.50},
                        {"fuse tea",2.5,"kekik","1 litre",0.0}
                };
//Menu functions
void welcome_menu(void);
void registration(void);
void login(void);
void user_menu(void);
int deposit_balance(void);
void account_remover(void);
void pastorders(void);
void edit_account(void);
void payment_methods(int ttl);
void fordelay(int j);
void loading();

//cart functions
int restaurant_pref(void);
int cart_generator(struct sb_data_som Som, struct Cart cart[5], int choice);
int cart_addons(struct sb_data_som Som, struct Cart cart[5], int purstyle);
int cart_displayer(struct sb_data_som Som, struct Cart cart[5], int purstyle);

#endif // SEKTEK_H_
```

IMPLEMENTATION

Explain the method applied for the problem solution

This program is based on the quick ordering of the user. First, it will register quickly and create a unique membership. Then the user selects one of the operations he wants to do in the menu. After making their choices in the food order menu, the total price to be paid is printed on the screen. Then he can switch to the payment method and pay from the wallet or instantly with cash. The change is not given in cash but transferred to your wallet instead.

TESTING

Briefly describe how you test your code

We did it by looking at the tests whether it opened the file, printed it, deleted it or not. In addition, if the same user exists and the password is less than 8 characters, We made it possible to make an error by looking at the tests.

We tested how the selected options lead the user.

In addition, in the ordering section, we checked if the options fill the structs properly. Finally, we tested whether the total displayer, which is the sum of the selected options, works correctly.

USER'S GUIDE

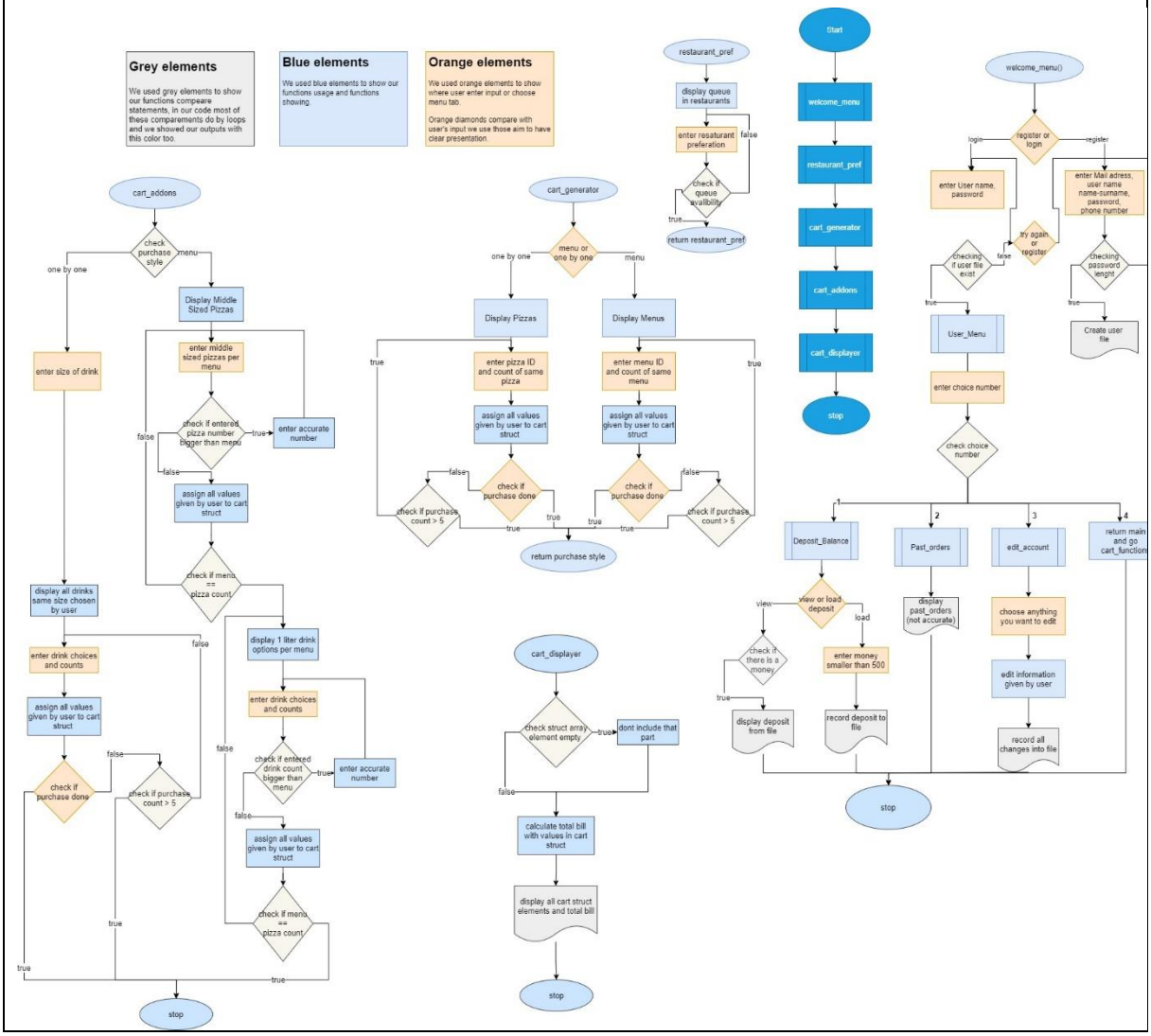
(Describe briefly how the user can use your program (input parameters of program, output of program, possible limitations, possible errors, etc.)

Users can easily register and log in from the options that appear. Then he can load the balance or place an order. Everything is in order of options, one after the other

Some rollback options do not work properly and stop the program.
Past order function not completed.

Cart_displayer function is strange as we don't understand. The user should only look at total.

GENERAL FLOWCHART OF THE PROJECT



CONCLUSION AND REMARKS

You can or can(not) put any remarks on your work. For example;
is your program works or not, if not why
which troubles during performance of project you faced with and how you overcome them;
is your program user friendly and how you can improve it;
what you achieved during performance of project; and so

There are two diffeerent version of our Project. Project without header runs without error. But the Project with header is not working at all. Because we wrote the entire of code with jobshare . Although the code we all write works without any problems, when we use the header, the code written using nested structures breaks. And that code is really big and very important part of our Project.

Deleting and renaming could not be done with the user file name assigned to the .txt extension with sprintf. So instead of opening an extra file and printing it and then deleting it, I created a single file again with **w** mode and entered the changed information in it. So the problem was solved. However, since remove function does not work in user delete function, I used the same method. Therefore, the file of the user whose information is deleted remains and another user with the same name cannot be opened.

Our program mostly user-friendly because its design is so simple.

We can improve it with a lot of extension. There are a lot of functions we cannot write due to lack of time and number of person in the Project. If we could complete the function of creating bill, writing to the user file, viewing the past order and finally logging in anonymously, we would have done everything in our mind.

REFERENCES

Put the list of references and sources (books, links to websites, videos, etc), which you used for project.

https://www.bilgigunlugum.net/prog/cprog/c_stdkut/stdio

https://stackoverflow.com/questions/13566082/how-to-check-if-a-file-has-content-or-not-using-c

https://stackoverflow.com/questions/61692985/create-a-simple-registration-and-login-system-with-c

https://www.programiz.com/c-programming/c-goto-statement

And other hundreds of stackoverflow pages that we don't know.

APPENDIX

PROJECT CODES

Put here code of your project.

No_header.c works without any problem.

no_header.txt

main.txt

sektek.h.txt

sektek.c.txt