**Computer Programming Assignment Report**

**1171100781 Faretz Emir**

**1171302426 Jamal Aljurayfani**

**Introduction**

The program commissioned is a program to act as a cash register machine for the restaurant “Mac C”. The program has two options, that is to add new food items and take orders. If (0) is input by the user, the program will quit altogether.

**Add new food items:**

Upon receiving user input in the form of an integer, (1), the program prompts the user for a description of the new food item to be added. Inputting (0) when prompted causes the program to return to the main menu. The section runs a loop that prompts the user for the description and price of the new food item until (0) is inputted. This data is then appended to an existing file, “Food.txt”.

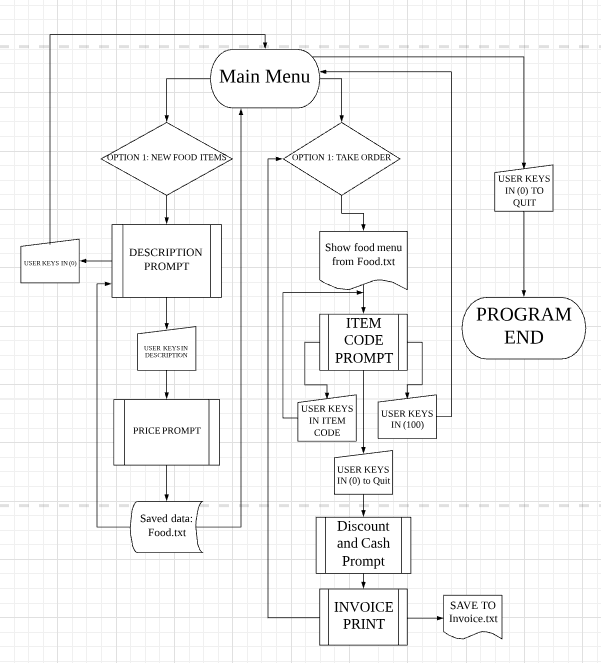
**Take order:**

The section of code for take order first displays the data from the file, “Food.txt” and prints it on the console. When prompted for the item code, if the user inputs (100), the program will revert to the main menu. The code runs a loop that consistently prompts the user for an item code from the list displayed on the console screen until (0) is inputted. Inputting (0) will cause the loop to end, prompt the user for a discount and cash tendered, and then print out an invoice. The same data for the invoice is then saved into a file, which is created if t does not yet exist, named “Invoice.txt”.

**Flow chart:**

A flow chart detailing the flow of the program is included in the next page.

**Flow Chart**



**Source code**

(My apologies for the messy indentations; I created the code on XUbuntu and brought it over to windows. A screenshot of the code in XUbuntu will be provided for better clarity.)

#include <stdio.h>#include <string.h>

int main(){

FILE \*fptr1, \*fptr2;

char item[1000], buffer[1000], \*string[100], zero[10]="0", f, \*sp, list[10000], list2[10000];

float price[1000]={10.40,9.45}, Tprice[1000], GTprice=0, Dprice, GGTprice, change;

int input, cd[100], p=0, x, r, s, t=0, code, Qt[1000], Q=0, R=0, add=2, dup=2, i, n=2, disc, cash;

do{

printf("WELCOME TO RESTAURANT MAC C - Main Menu\n");

printf("[1] Add new food items\n[2] Take order\n");

printf("Enter ITEM CODE (0 to Quit) :\n");

scanf("%d", &input);

if(input==0)

{return 0;}

else if(input==1)

{

fptr1 = fopen("Food.txt","r");

f = fgetc(fptr1);

while(f != EOF)

{

printf("%c", f);

f = fgetc(fptr1);

}

fclose(fptr1);

do{

fptr1 = fopen("Food.txt","a+");

printf("\nEnter a description (0 to Main Menu): \n");

scanf("%s", item);

string[dup] = strdup(item);

dup++;

x = strcmp(zero,item);

if(x==0)

break;

printf("Enter a price: \n");

scanf("%f", &price[n]);

add++;

sprintf(list,"\n%d\t\t%s\t\t%.2f",add, item, price[n]);

n++;

printf("\n%s\n", list);

fputs(list, fptr1);

fclose(fptr1);

} while(x != 0);

}

else if(input==2)

{

do{

fptr1 = fopen("Food.txt","r");

r = fgetc(fptr1);

while(r != EOF)

{

printf("%c", r);

r = fgetc(fptr1);

}

fclose(fptr1);

do{

printf("\nEnter ITEM CODE (0 to Quit, 100 for Main Menu)\n");

scanf("%d", &code);

if(code == 0)

{

printf("\nEnter Discount (PERCENT)\t\t:");

canf("%d", &disc);

printf("\t\t\tINVOICE RESTAURANT MAC C\t\t\t\n");

printf("--------------------------------------------------------------------------------\n");

printf("ITEM CODE\tDESCRIPTION\tPRICE(RM)\tQUANTITY\tTOTAL(RM)\n");

for(R=0;R<add;R++)

{

GTprice = GTprice + Tprice[R];

Dprice = GTprice \* disc / 100;

GGTprice = GTprice - Dprice;

}

for(R=0;R<add;R++)

printf("%d\t\t%s\t\t%.2f\t\t%d\t\t%.2f\n", cd[R], string[R], price[R], Qt[R],Tprice[R]);

printf("--------------------------------------------------------------------------------\n");

printf("Subtotal before Discount\t\t %.2f\n", GTprice);

printf("Discount %d(PERCENT)\t\t\t- %.2f\n", disc, Dprice);

printf("--------------------------------------------------------------------------------\n");

printf("TOTAL\t\t\t%.2f\n\n", GGTprice);

printf("CASH TENDERED\t\t\t:");

scanf("%d", &cash);

if(cash<GGTprice)

{

printf("INSUFFICIENT CASH!");

scanf("%d", &cash);

}

change = cash - GGTprice;

printf("CHANGE\t\t\t: %.2f\n", change);

printf("PRINTING OUT INVOICE\n");

fptr2 = fopen("Invoice.txt", "a+");

fprintf(fptr2,"\t\t\tINVOICE RESTAURANT MAC C\t\t\t\n");

fprintf(fptr2,"\n--------------------------------------------------------------------------------\n");

fprintf(fptr2,"\nITEM CODE\tDESCRIPTION\tPRICE(RM)\tQUANTITY\tTOTAL(RM)\n");

for(R=0;R<add;R++)

fprintf(fptr2,"%d\t\t%s\t\t%.2f\t\t%d\t\t%.2f\n", cd[R], string[R], price[R], Qt[R], Tprice[R]);

fprintf(fptr2,"--------------------------------------------------------------------------------\n");

fprintf(fptr2,"Subtotal before Discount\t\t %.2f\n", GTprice);

fprintf(fptr2,"Discount %d(PERCENT)\t\t\t- %.2f\n", disc, Dprice);

fprintf(fptr2,"--------------------------------------------------------------------------------\n");

fprintf(fptr2,"TOTAL\t\t\t%.2f\n\n", GGTprice);

fprintf(fptr2,"CASH TENDERED\t\t\t:\t\t%d\n", cash);

fprintf(fptr2,"CHANGE\t\t\t: %.2f\n", change);

fclose(fptr2);

}

else if(code <= add && code>0)

{

cd[code-1] = code;

if(code == 1)

{ code = cd[code-1];

sp = buffer;

sp ="BigM";

string[0] = sp;}

else if(code == 2)

{ code = cd[code-1];

sp = buffer;

sp ="Cheeseburger";

string[1] = sp;}

Qt[code-1]++;

t++;

Tprice[code-1] = price[code-1] \* Qt[code-1];

}

else if(code == 100)

{break;}

else if(code != 100 && code != 0)

printf("ERROR, NO CODE!!");

}while(code != 100);

}while(code != 100 && code != 0);

if(code==0)

{break;}

}

else

{ printf("ERROR! NO CODE!\n");

printf("TRY AGAIN!\n");

scanf("%d", &input);

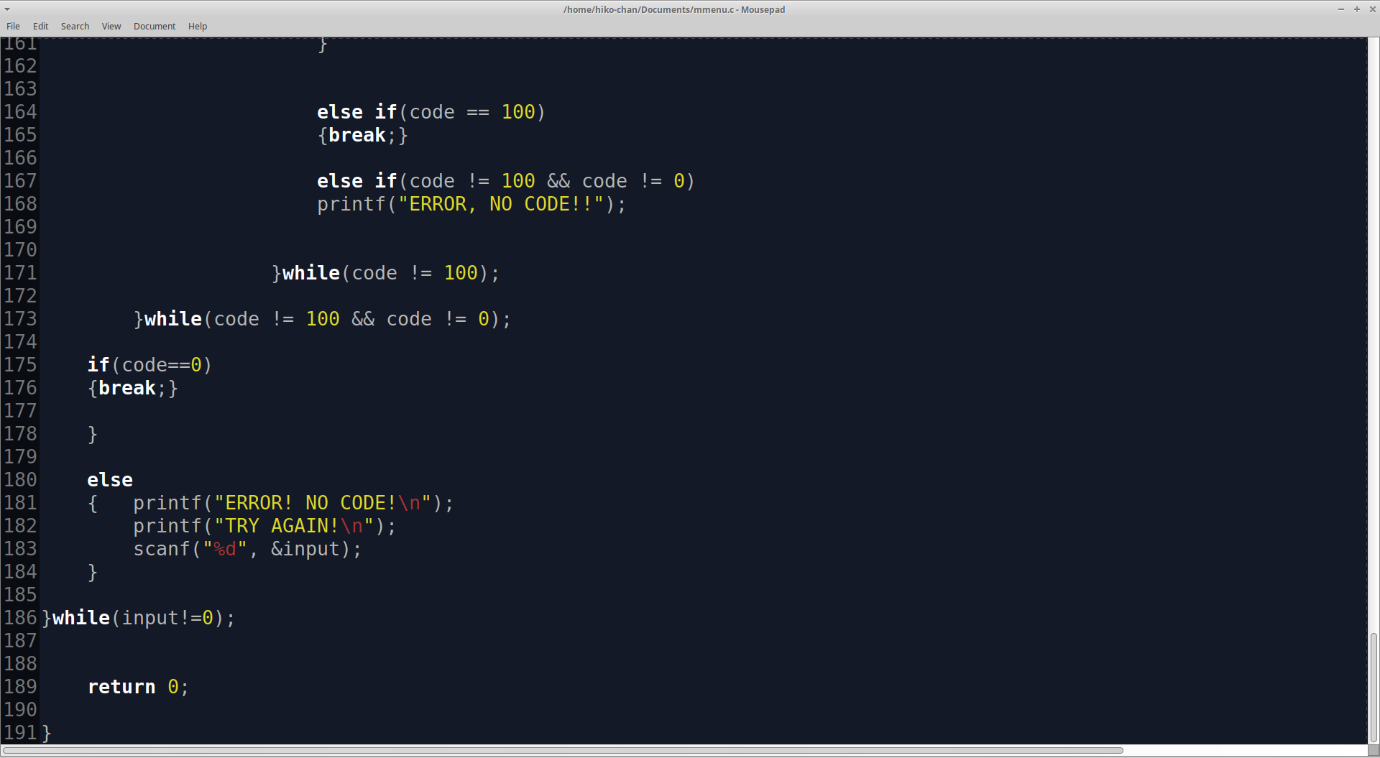
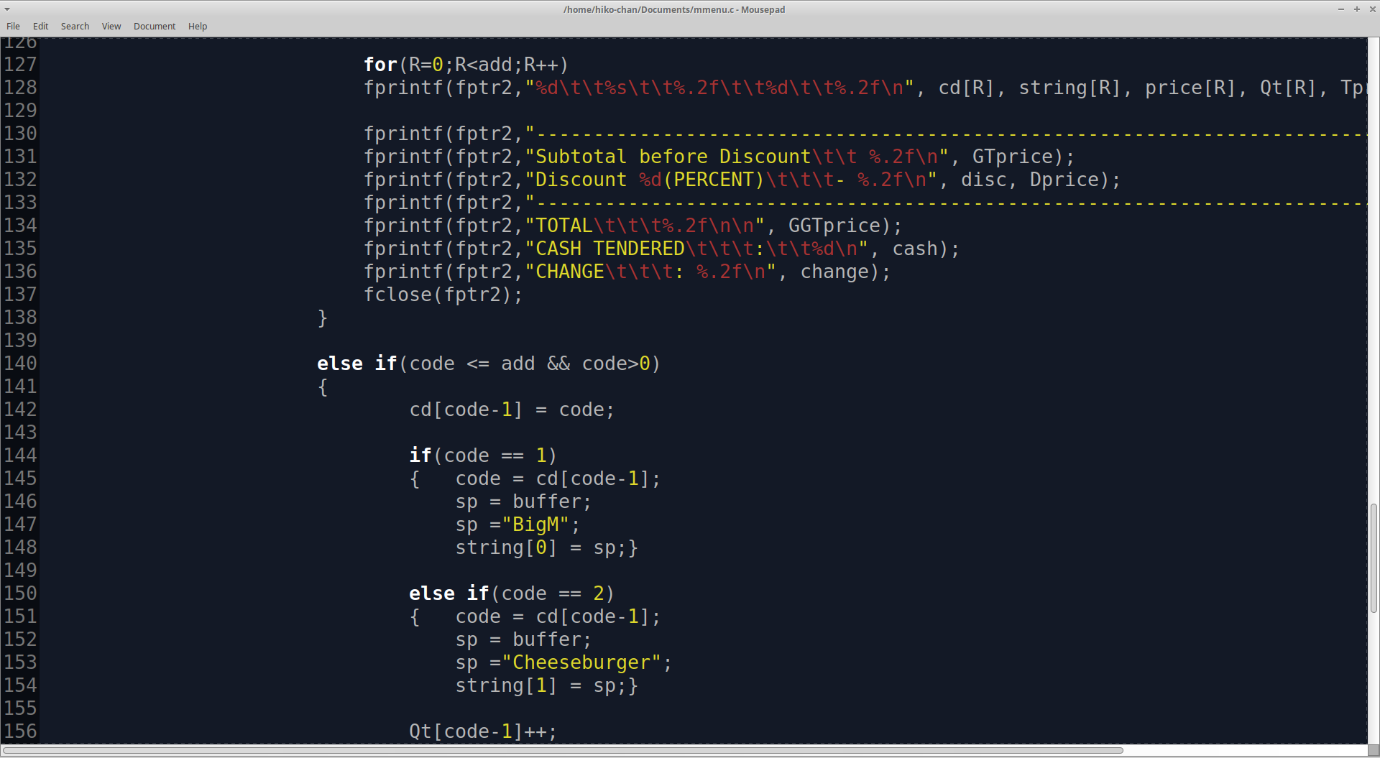
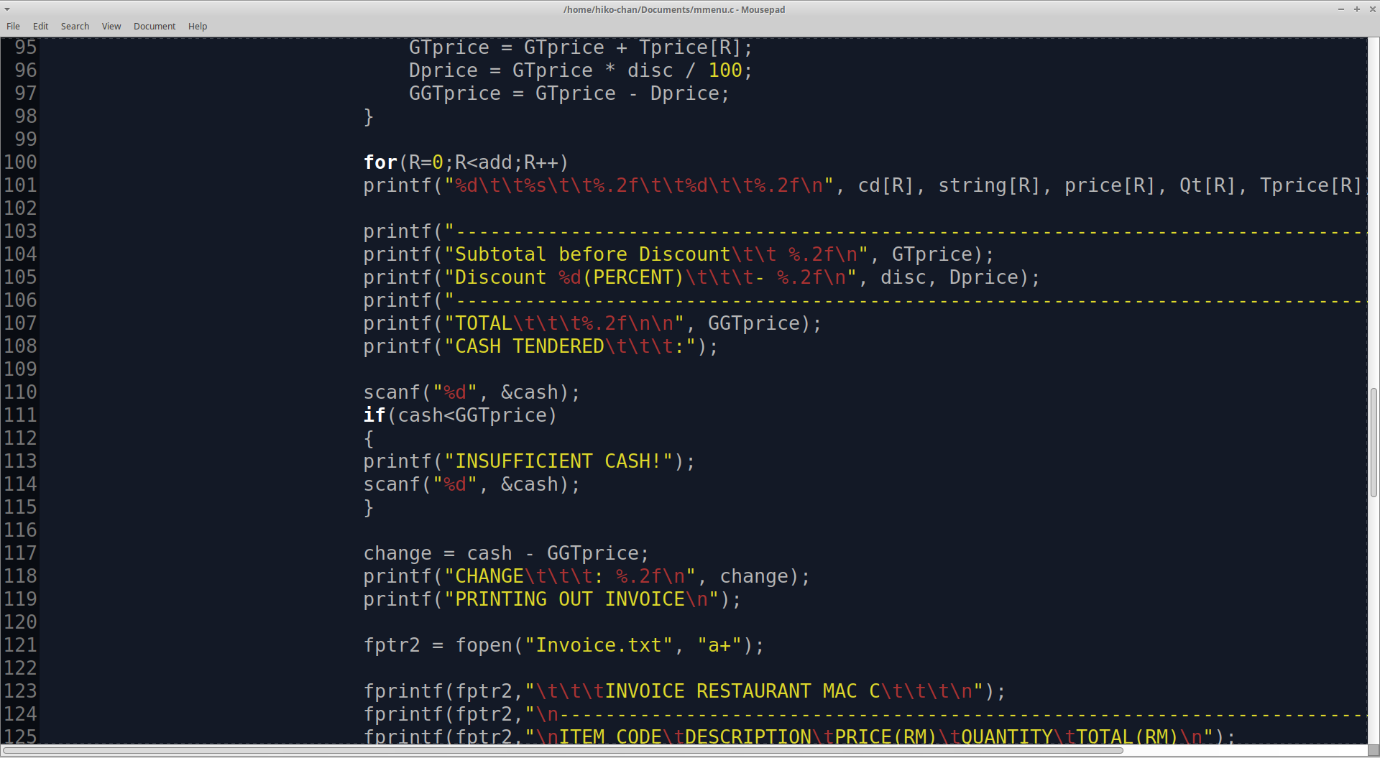
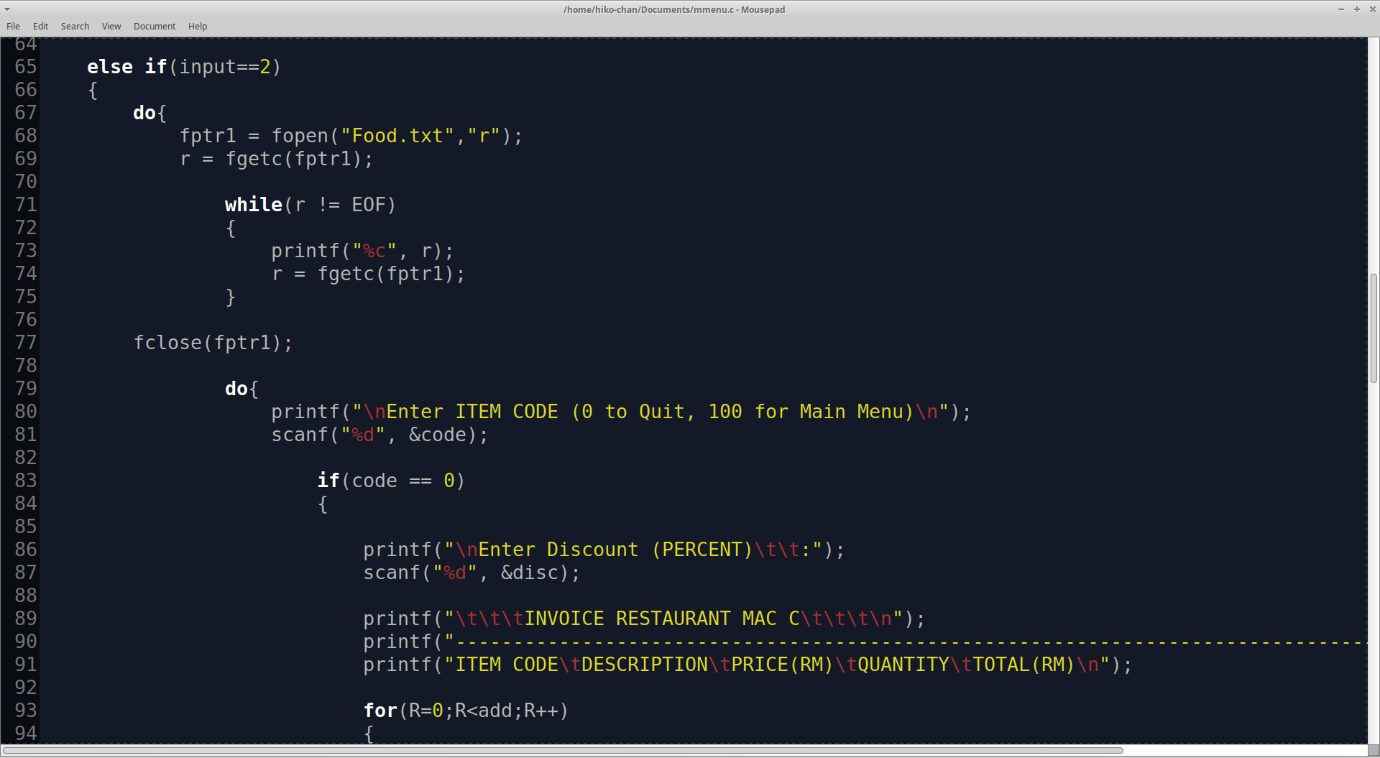
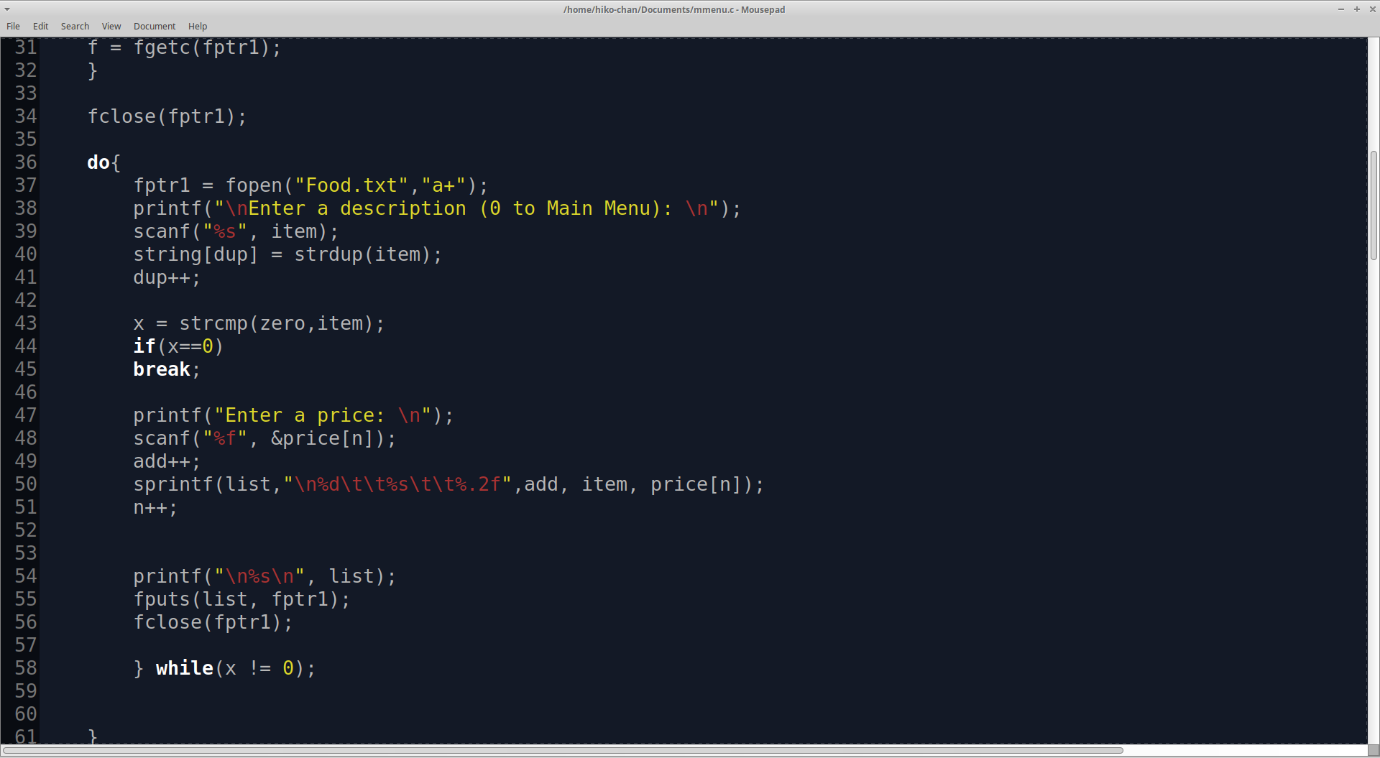
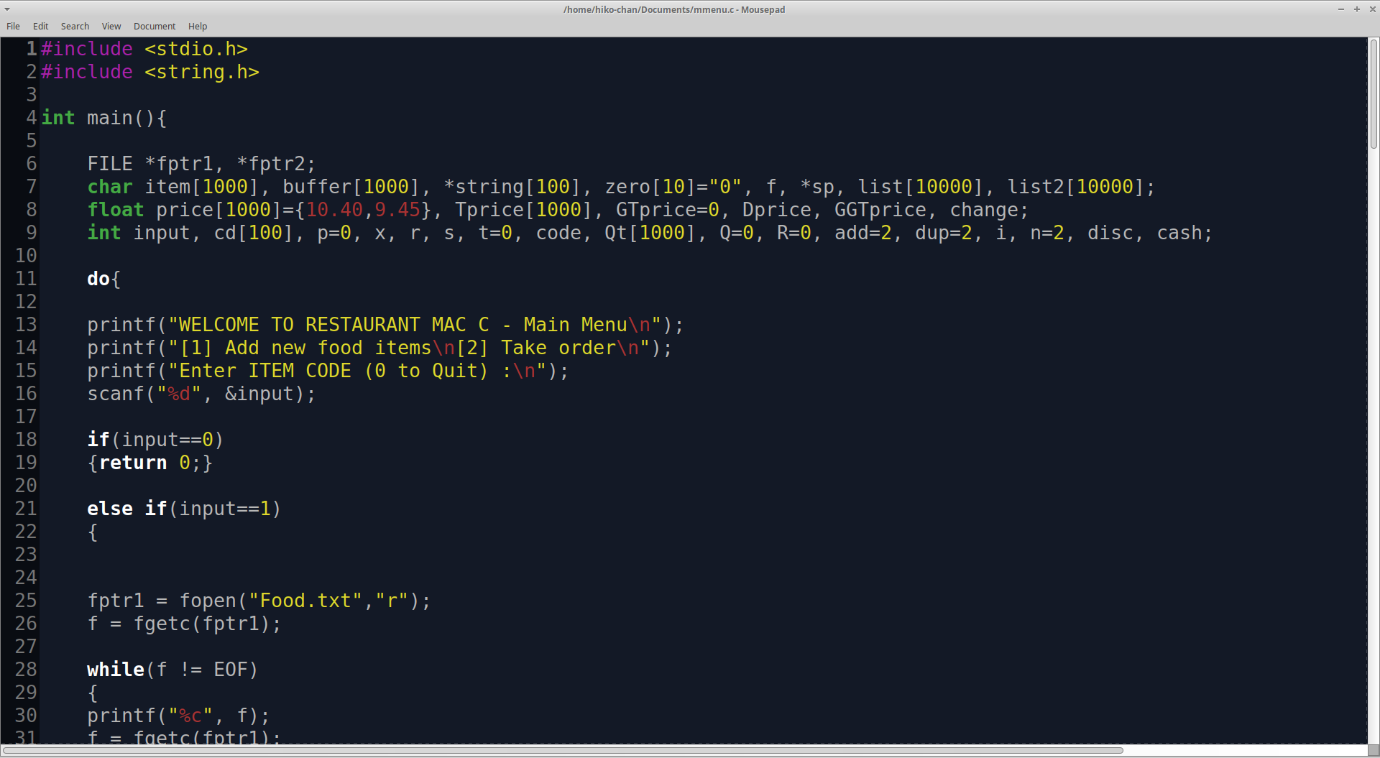
}

}while(input!=0);

return 0;

}

**Screenshots:**

****

**Program Comments:**

The program is simple in its flow, yet complex to code. Some complications were met, especially when dealing with pointers, files and arrays.

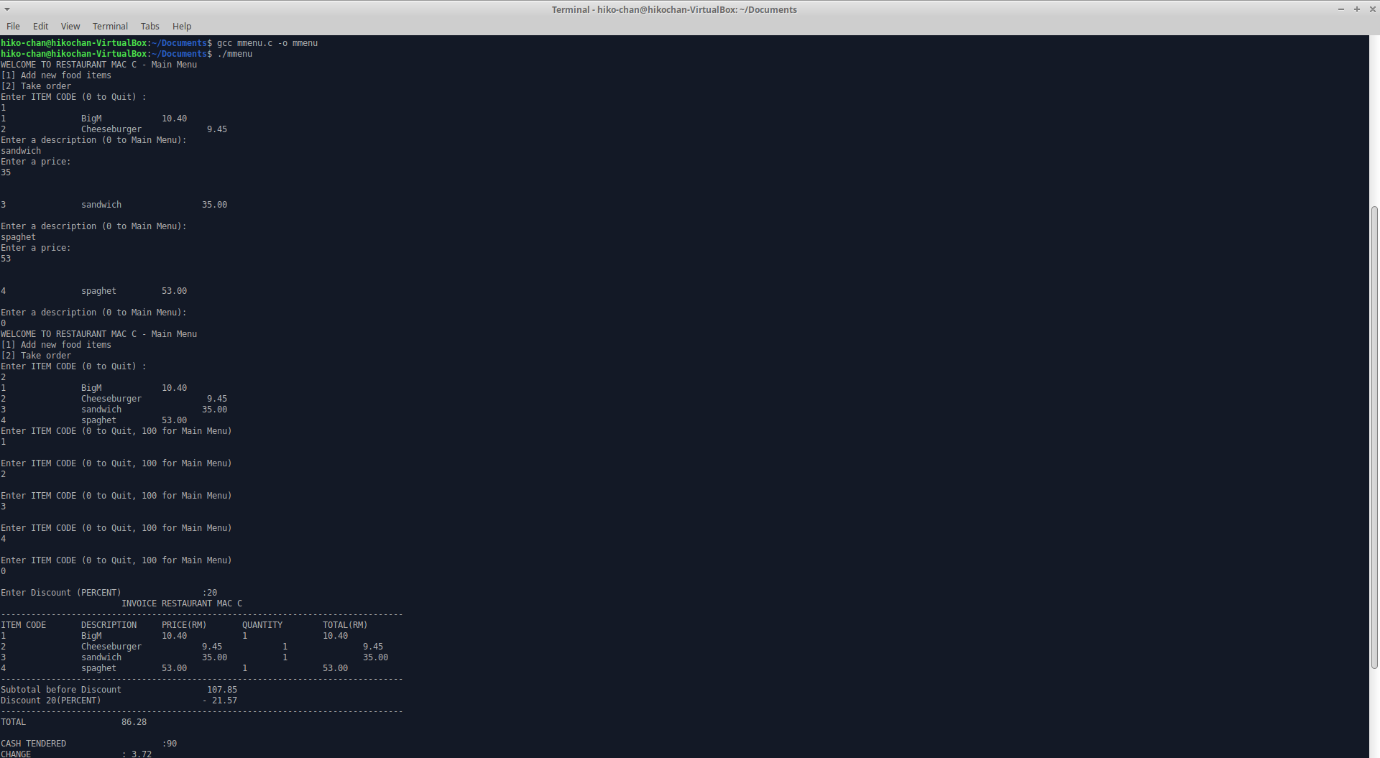
Some of the complications include but are not limited to:

* Dealing with the loops and how to get back from one option to the main menu and so on.
* Figuring out how to pass variables from the command line to a text file
* Obtaining said variables for the text file and saving them as strings in an array
* Passing strings to an array and then printing them out on the console.
* Figuring out the sequence of the code so that no sections are redundant
* Dealing with variables in different forms of variable types such as characters, integers and floats in such a way as to not complicate the workload

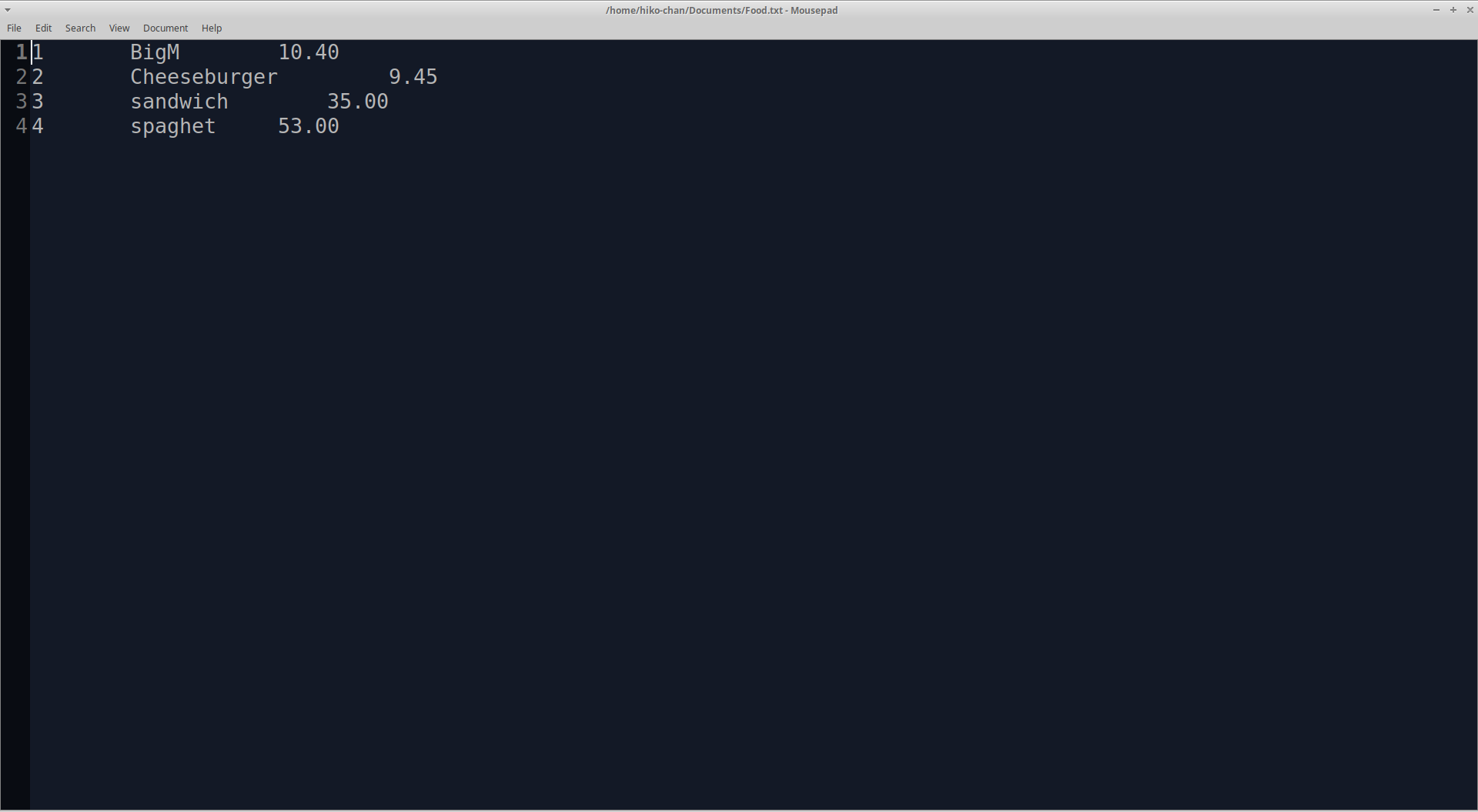
**Discussion**

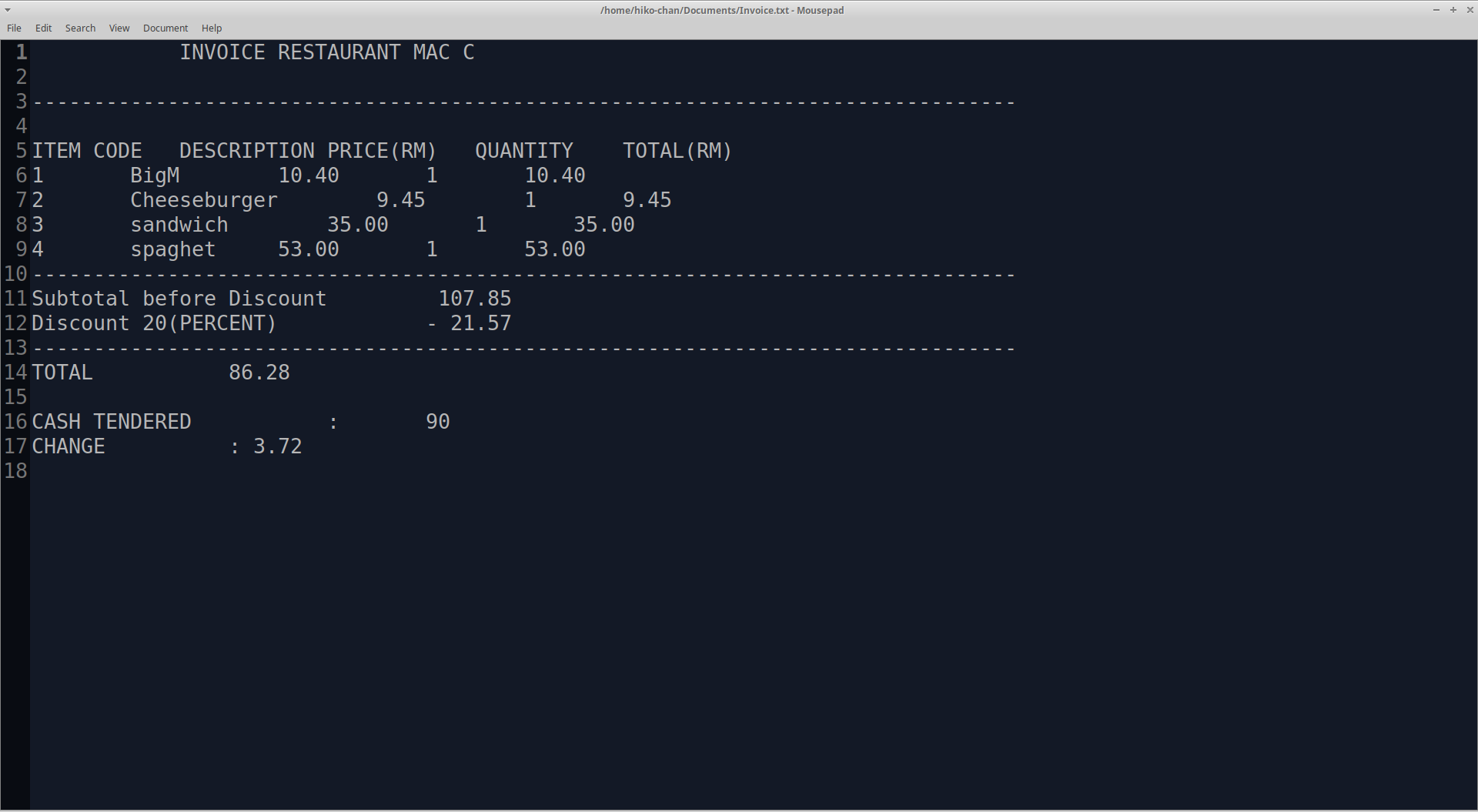
* The program is not one that can be dealt with in a single day, let alone in a period of a few hours; Time management is key to programming
* Various functions and variable types to be used must be planned out beforehand to lighten the workload
* A flowchart makes it easier to see the flow of the program and determine the functions needed to be used
* Arrays and pointers must be dealt with in a proper manner; they cannot simply be treated as a one dimensional variable.

**Results**

**Running the program: **

**Food.txt**

****

**Invoice.txt**

**Discussion**

The program runs well without any discernable errors upon compilation. The program was written, run and compiled on XUbuntu.

The program runs according to the flowchart. The end result is a display of the processes on the console screen, ending with an invoice printout before the program exits.

**Main Menu**

The upon running, the user is greeted with a display of the options to choose from, namely “enter new food items” and “take order”, followed by a prompt for user input in the form of an integer corresponding to the number (#) of the options presented. If the user inputs (0) in the main menu, the program quits.

**New food items**

Upon selecting new food items, the user is presented with a list of existing food items followed by a prompt for user input for description (of the food item), and price of the same food item. The code is then looped to ask for the same variables until the user inputs (0) to bring the user back to the main menu. The data input is then stored and appended in an existing text file, “Food.txt”.

**Take order**

The user is shown a food menu, retrieved from “Food.txt”. The program then runs a loop which prompts the user for an input of the item code(Until the user inputs(100) which takes them back to the main menu, or when (0) is input to quit the loop), while cataloging the quantity and item code to two separate arrays.

Upon quitting the loop, the user is then prompted for a discount percentage and cash tendered, then is presented with a print of the invoice on the console screen. This same invoice is then saved in a text file named “Invoice.txt”.

Afterwards, the program is taken back to the “Take Order” portion of the code. To quit the program altogether, the user has to exit to the main menu, and then input (0) to close the program.