AIM :- To show how a vending machine works

Functions

Function 1 :- Welcome
It welcomes the user to vending machine.

Function 2 :- Rules
It tell the use what not to do to vending machine.
It tell that the user can buy a single type of product at a time.

Function 3 :- Snacks
This function is called when the user press 1 for choosing snacks.In this function it prints the name of the snacks available in the vending machine with the price and weight of the snack and a special code is given to every variety of snack. After printing the available snacks it calls selction_snack function and it stores its value in a variable called amount.

Function 4 :- Selection_snack
In this function it first ask the code and quantity of the desired product and after that if else condition is runned on the basis of code entered. If the code entered is matched with the product code it tells the total price of the product and if the code is not matched with any of the if condition else condition is runned and in it a goto statement is used with directly takes the user to the line where the user was asked to enter the code of the product. After the matching of the code with if condition it calculates the total price of the product by multiplying the quantity of the product to price of the product.This Function giver total_amount as the return value.This value is stored on the variable amount in snack function.

Function 5:- Drinks
If the user press 2 in the selection option.He will be directed to drinks function.In this function all the available drinks in vending machine is displayed to the user.Every drink is given a special code.Price and volume of the drink is also displayed.After the displayed of the information of the drinks selection_drinks function is called and its value is stored in variable amount.

Function 6:- Selection_drinks
In this function it first ask the code and quantity of the desired product and after that if else condition is runned on the basis of code entered. If the code entered is matched with the product code it tells the total price of the product and if the code is not matched with any of the if condition else condition is runned and in it a goto statement is used with directly takes the user to the line where the user was asked to enter the code of the product. After the matching of the code with if condition it calculates the total price of the product by multiplying the quantity of the product to price of the product.This Function giver total_amount as the return value.This value is stored on the variable amount in drink function.

Function 7 :- Deposit
This function basically takes money deposit from the user. it returns the money in the main function from where it was called and the returned amount is store in the variable named as cash.

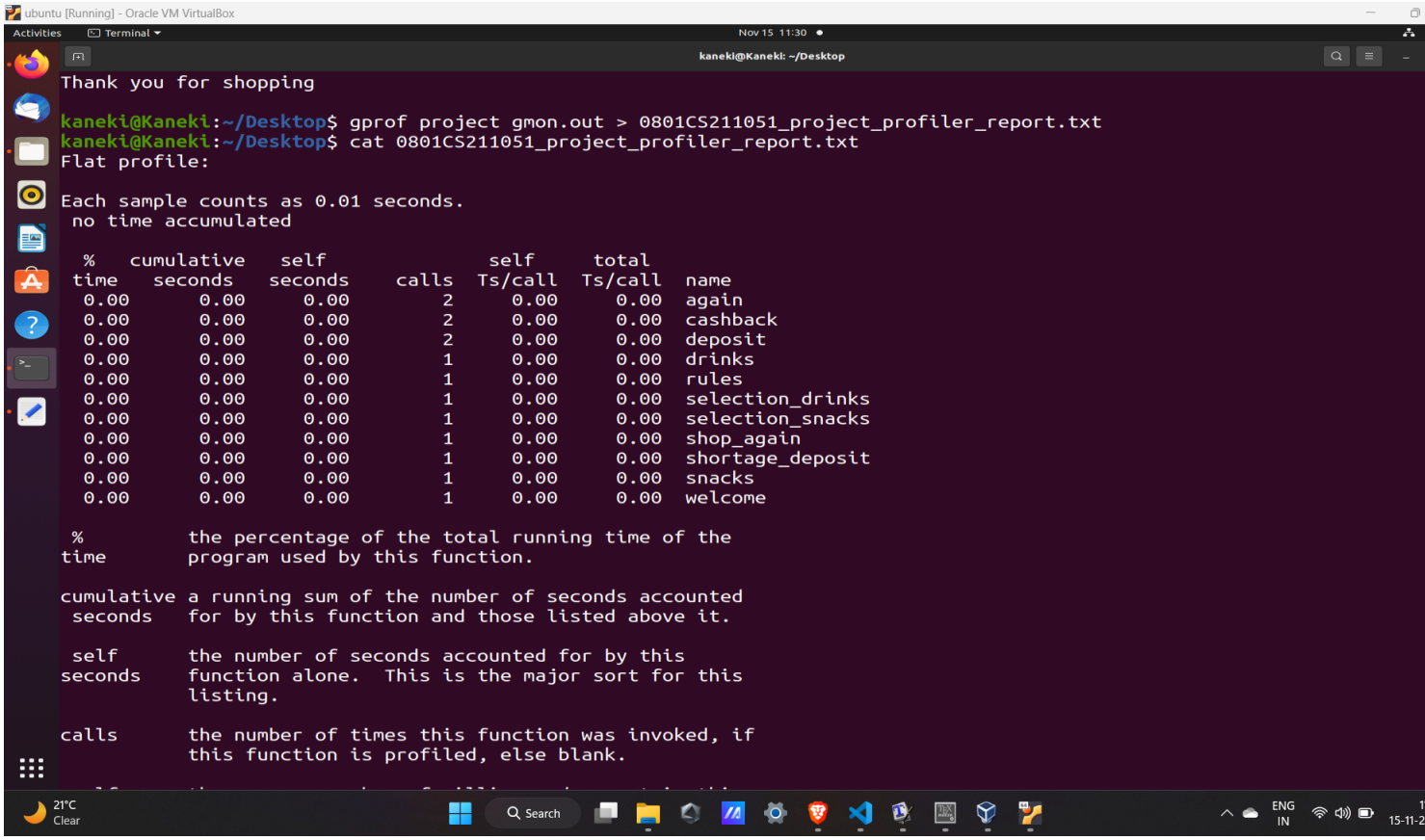
Function 8 :- Cashback
It take one integer type datatype as a parameter which is cash.This function basically calculates that your entered amount is is sufficient or not if it is not sufficient shortage_deposit is called and its value is stored in the valuable remaining,This function gives remaining*-1 as return value.

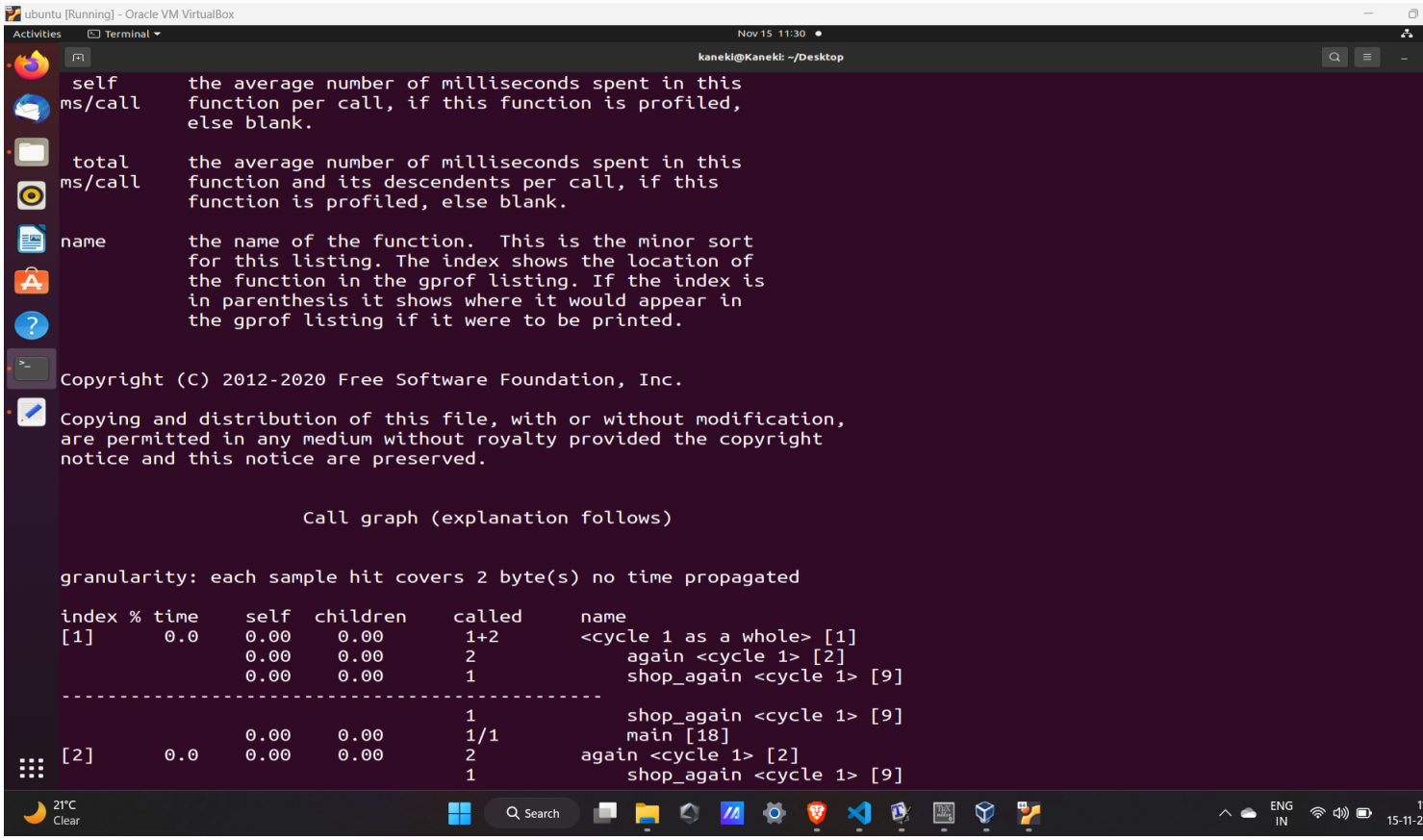
Function 9:- Shortage_deposit
this function basically tells the user that you are short on ___ amount of money and then ask user to input more money. Even after giving more money if its lesser then the amount shortage_deposit function is called again like in recursion and if the total cash is larger then the amount of product then remaining is given as returned value.

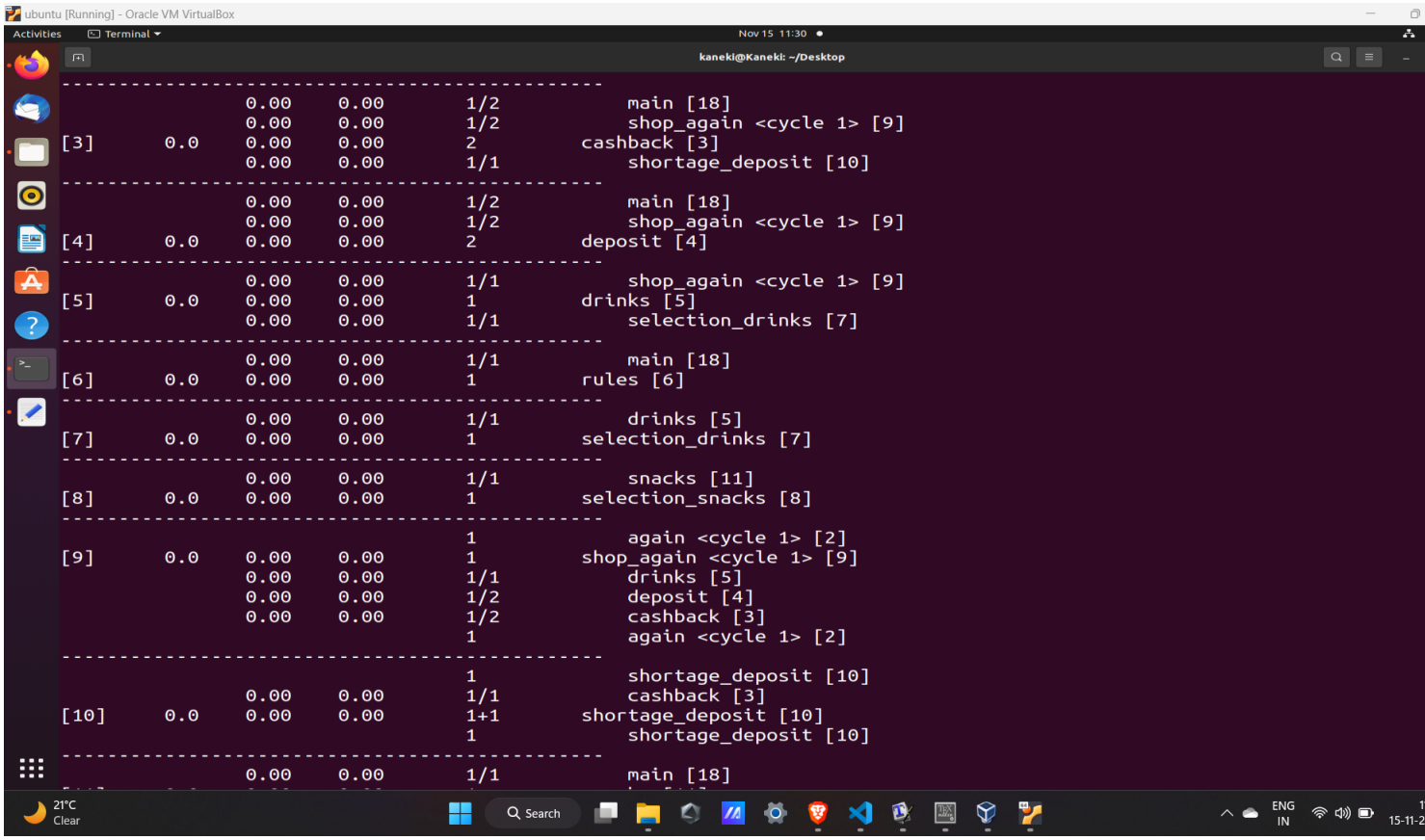
Function 10:- Again
This function ask the user to press 1 if user want to buy something else and press 2 to quit. If the user presses 1 then shop_again function is called and if the user press 2 then thanks message is printed and on entering any other value it will print you pressed wrong number and a goto statement will take you to give the response again.

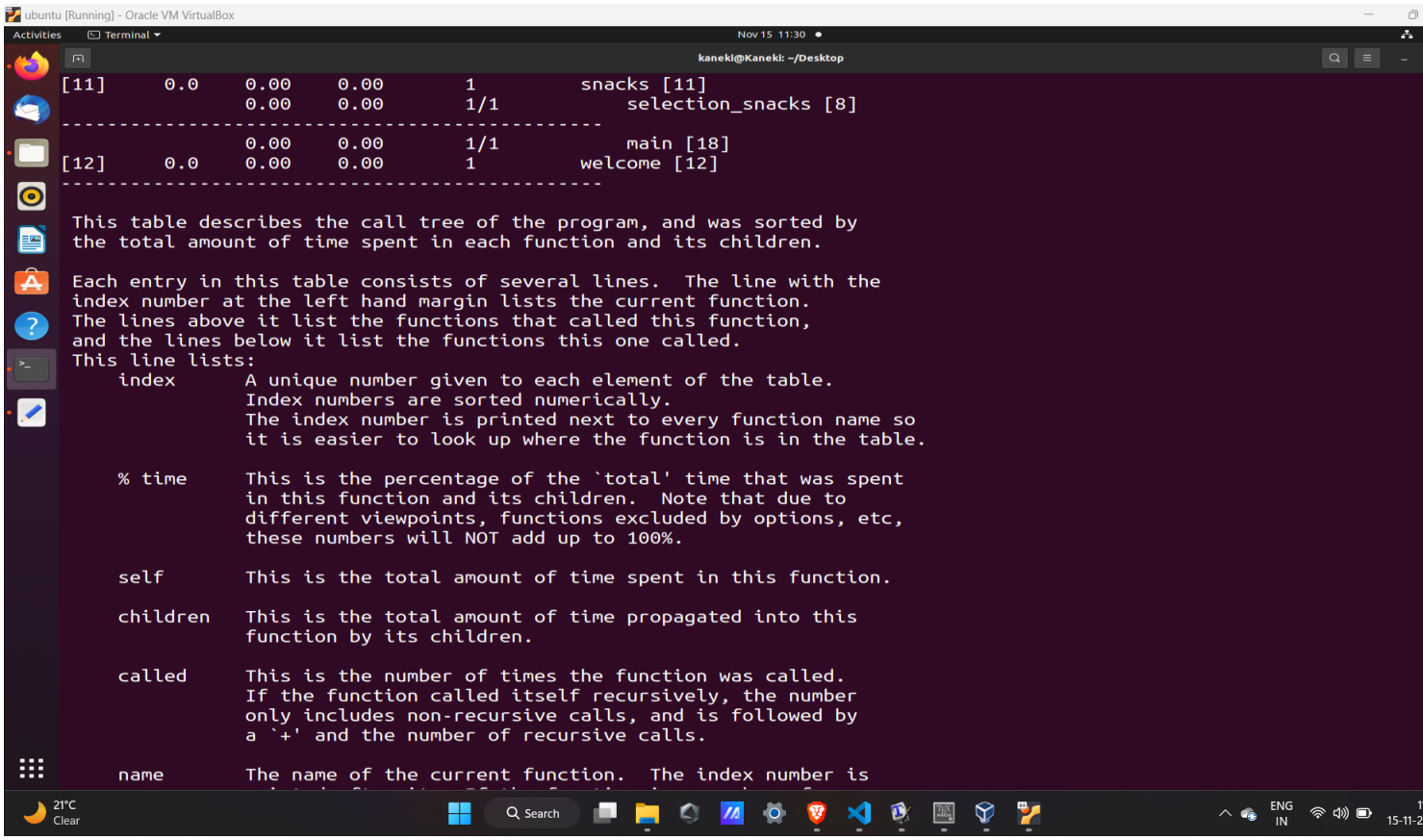
Function 11:- Shop_again
this function basically calls all the required function in a sequential way. It does not print the welcome and rules message.

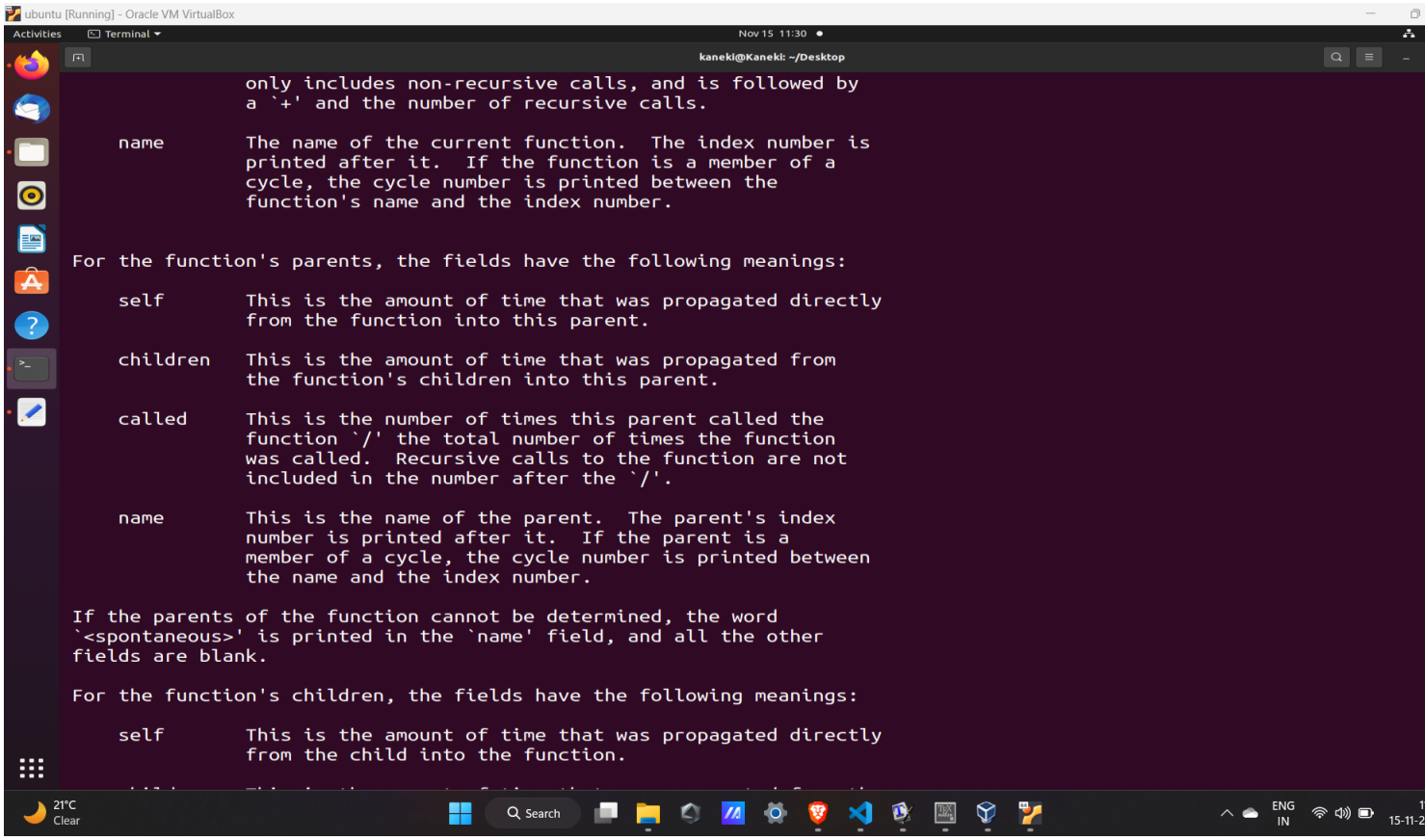
Profiling Report

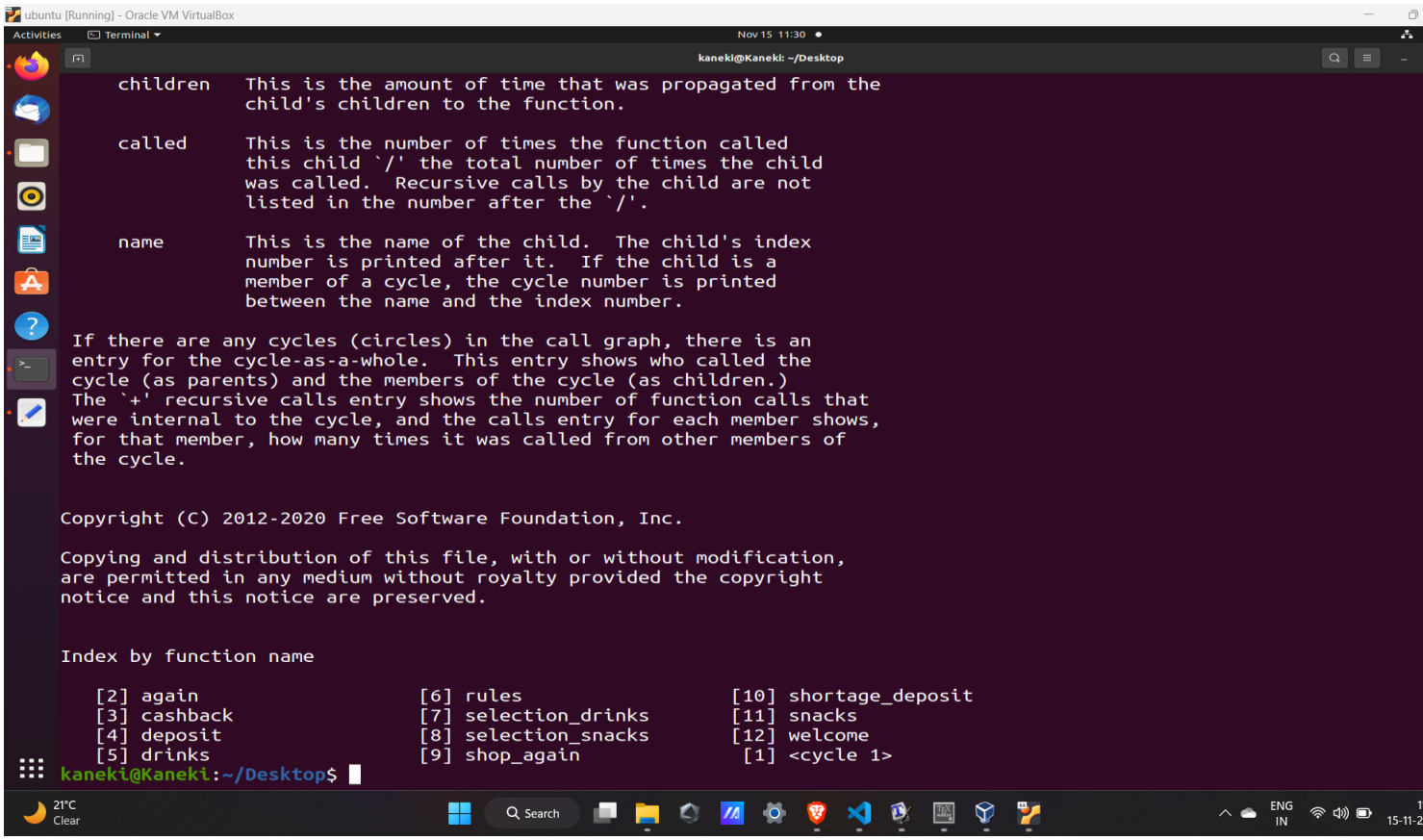




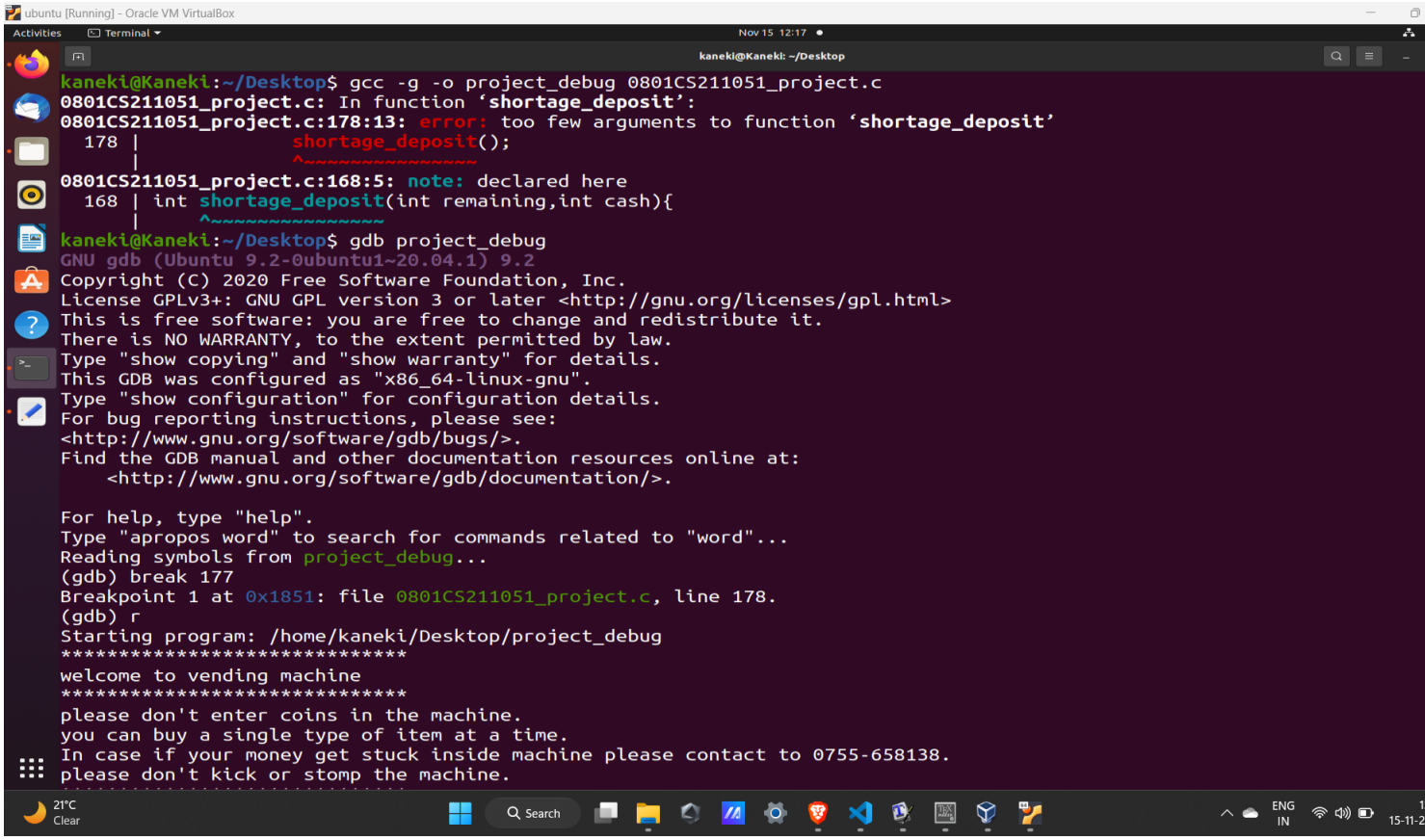

ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 15 11:30
kaneki@Kaneki: ~/Desktop
0.00 0.00 1/2 main [18]
0.00 0.00 1/2 shop_again <cycle 1> [9]
[3] 0.0 0.00 0.00 2 cashback [3]
0.00 0.00 1/1 shortage_deposit [10]
0.00 0.00 1/2 main [18]
0.00 0.00 1/2 shop_again <cycle 1> [9]
[4] 0.0 0.00 0.00 2 deposit [4]
0.00 0.00 1/1 shop_again <cycle 1> [9]
[5] 0.0 0.00 0.00 1 drinks [5]
0.00 0.00 1/1 selection_drinks [7]
0.00 0.00 1/1 main [18]
[6] 0.0 0.00 0.00 1 rules [6]
0.00 0.00 1/1 drinks [5]
[7] 0.0 0.00 0.00 1 selection_drinks [7]
0.00 0.00 1/1 snacks [11]
[8] 0.0 0.00 0.00 1 selection_snacks [8]
1 again <cycle 1> [2]
[9] 0.0 0.00 0.00 1 shop_again <cycle 1> [9]
0.00 0.00 1/1 drinks [5]
0.00 0.00 1/2 deposit [4]
0.00 0.00 1/2 cashback [3]
1 again <cycle 1> [2]
1 shortage_deposit [10]
0.00 0.00 1/1 cashback [3]
[10] 0.0 0.00 0.00 1+1 shortage_deposit [10]
1 shortage_deposit [10]
0.00 0.00 1/1 main [18]
21°C
Clear
Search
ENG
IN

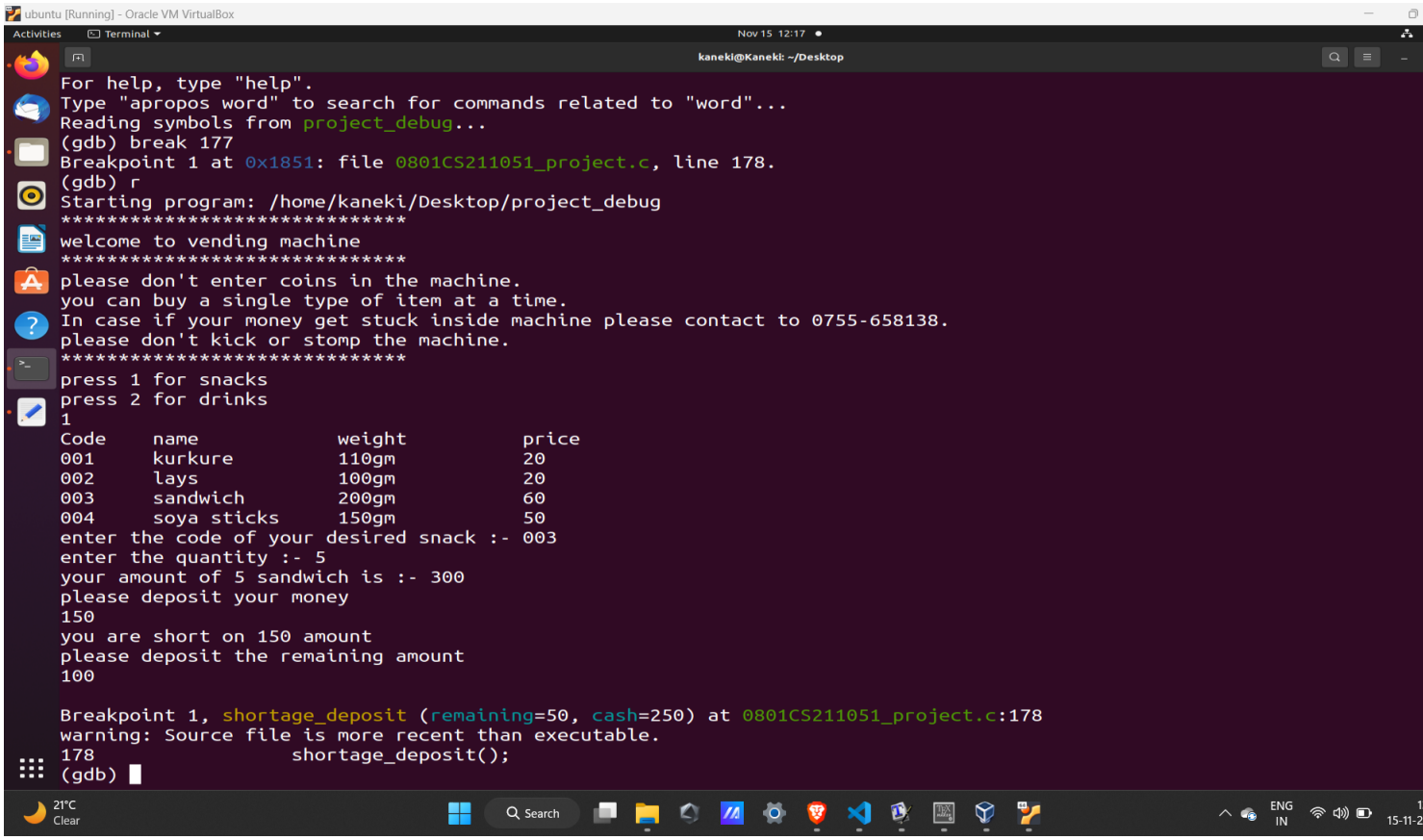

ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 15 11:30
kaneki@Kaneki: ~/Desktop
[11] 0.0 0.00 0.00 1 snacks [11]
0.00 0.00 1/1 selection_snacks [8]
0.00 0.00 1/1 main [18]
[12] 0.0 0.00 0.00 1 welcome [12]
This table describes the call tree of the program, and was sorted by
the total amount of time spent in each function and its children.
Each entry in this table consists of several lines. The line with the
index number at the left hand margin lists the current function.
The lines above it list the functions that called this function,
and the lines below it list the functions this one called.
This line lists:
index A unique number given to each element of the table.
Index numbers are sorted numerically.
The index number is printed next to every function name so
it is easier to look up where the function is in the table.
% time This is the percentage of the `total' time that was spent
in this function and its children. Note that due to
different viewpoints, functions excluded by options, etc,
these numbers will NOT add up to 100%.
self This is the total amount of time spent in this function.
children This is the total amount of time propagated into this
function by its children.
called This is the number of times the function was called.
If the function called itself recursively, the number
only includes non-recursive calls, and is followed by
a `+' and the number of recursive calls.
name The name of the current function. The index number is
21°C
Clear
Search
ENG
IN


ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 15 11:30
kaneki@Kaneki: ~/Desktop
only includes non-recursive calls, and is followed by
a `+' and the number of recursive calls.
name The name of the current function. The index number is
printed after it. If the function is a member of a
cycle, the cycle number is printed between the
function's name and the index number.
For the function's parents, the fields have the following meanings:
self This is the amount of time that was propagated directly
from the function into this parent.
children This is the amount of time that was propagated from
the function's children into this parent.
called This is the number of times this parent called the
function `/' the total number of times the function
was called. Recursive calls to the function are not
included in the number after the `/'.
name This is the name of the parent. The parent's index
number is printed after it. If the parent is a
member of a cycle, the cycle number is printed between
the name and the index number.
If the parents of the function cannot be determined, the word
`<spontaneous>' is printed in the `name' field, and all the other
fields are blank.
For the function's children, the fields have the following meanings:
self This is the amount of time that was propagated directly
from the child into the function.
21°C
Clear
Search
ENG
IN


ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 15 11:30
kaneki@Kaneki: ~/Desktop
children This is the amount of time that was propagated from the
child's children to the function.
called This is the number of times the function called
this child `/' the total number of times the child
was called. Recursive calls by the child are not
listed in the number after the `/'.
name This is the name of the child. The child's index
number is printed after it. If the child is a
member of a cycle, the cycle number is printed
between the name and the index number.
If there are any cycles (circles) in the call graph, there is an
entry for the cycle-as-a-whole. This entry shows who called the
cycle (as parents) and the members of the cycle (as children.)
The `+' recursive calls entry shows the number of function calls that
were internal to the cycle, and the calls entry for each member shows,
for that member, how many times it was called from other members of
the cycle.
Copyright (C) 2012-2020 Free Software Foundation, Inc.
Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.
Index by function name
[2] again
[3] cashback
[4] deposit
[5] drinks
[6] rules
[7] selection_drinks
[8] selection_snacks
[9] shop_again
[10] shortage_deposit
[11] snacks
[12] welcome
[1] <cycle 1>
kaneki@Kaneki:~/Desktop$
21°C
Clear
Search
ENG
IN

Debugging


ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 15 12:17
kaneki@Kaneki: ~/Desktop
kaneki@Kaneki:~/Desktop$ gcc -g -o project_debug 0801CS211051_project.c
0801CS211051_project.c: In function ‘shortage_deposit’:
0801CS211051_project.c:178:13: error: too few arguments to function ‘shortage_deposit’
178 | shortage_deposit();
| ^~~~~~~~~~~~~~~~
0801CS211051_project.c:168:5: note: declared here
168 | int shortage_deposit(int remaining,int cash){
| ^~~~~~~~~~~~~~~~
kaneki@Kaneki:~/Desktop$ gdb project_debug
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from project_debug...
(gdb) break 177
Breakpoint 1 at 0x1851: file 0801CS211051_project.c, line 178.
(gdb) r
Starting program: /home/kaneki/Desktop/project_debug
****
welcome to vending machine
****
please don't enter coins in the machine.
you can buy a single type of item at a time.
In case if your money get stuck inside machine please contact to 0755-658138.
please don't kick or stomp the machine.
21°C
Clear
Search
ENG
IN


ubuntu [Running] - Oracle VM VirtualBox
Activities
Terminal
Nov 15 12:17
kaneki@Kaneki: ~/Desktop
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from project_debug...
(gdb) break 177
Breakpoint 1 at 0x1851: file 0801CS211051_project.c, line 178.
(gdb) r
Starting program: /home/kaneki/Desktop/project_debug
****
welcome to vending machine
****
please don't enter coins in the machine.
you can buy a single type of item at a time.
In case if your money get stuck inside machine please contact to 0755-658138.
please don't kick or stomp the machine.
****
press 1 for snacks
press 2 for drinks
1
Code name weight price
001 kurkure 110gm 20
002 lays 100gm 20
003 sandwich 200gm 60
004 soya sticks 150gm 50
enter the code of your desired snack :- 003
enter the quantity :- 5
your amount of 5 sandwich is :- 300
please deposit your money
150
you are short on 150 amount
please deposit the remaining amount
100
Breakpoint 1, shortage_deposit (remaining=50, cash=250) at 0801CS211051_project.c:178
warning: Source file is more recent than executable.
178 shortage_deposit();
(gdb)
21°C
Clear
Search
ENG
IN

Code in C Language

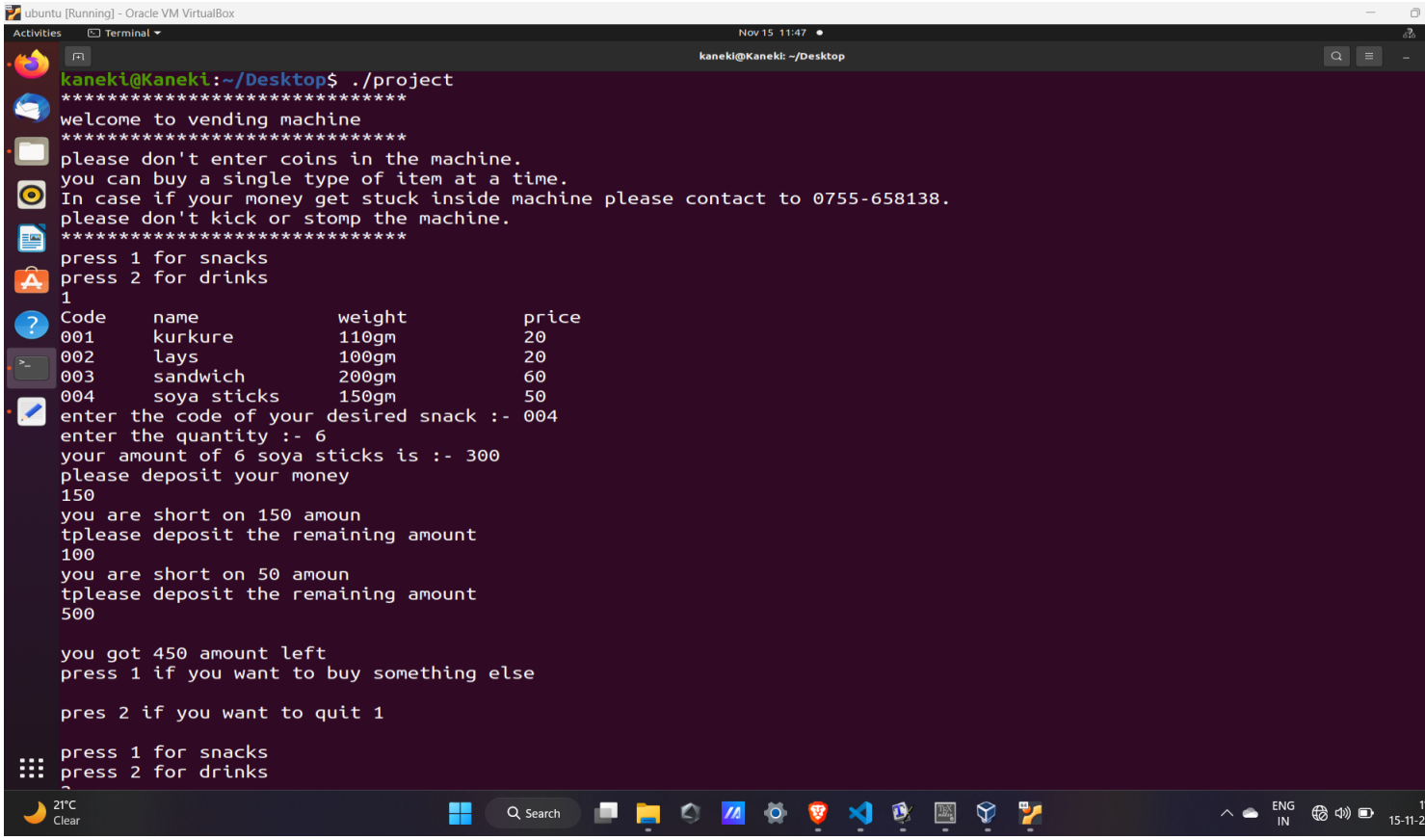
```
#include⟨stdio.h⟩ int amount,purse; //declared global variables
void welcome(); //declaring welcome function
void rules(); //declaring rules function
void snacks(); //declaring snacks function
void drinks(); //declaring drinks function
int selection_snacks(); //declaring selection_snacks function
int selection_drinks(); //declaring selection_drinks function
int deposit(); //declaring deposit function
int cashback(int); //declaring cashback function
int shortage_deposit(int,int); //declaring shortage_deposit function
void again(); //declaring again function
void shop_again(); //declaring shop_again function

int main(){
    int selection,cash;
    welcome();
    rules();
    again:
    printf("\n press 1 for snacks \n press 2 for drinks \n");
    scanf("%d",&selection);
    switch (selection)
//selection for whether user want snacks or drinks
    {
    case 1:
        snacks();
//snacks function is called
        break;
    case 2:
        drinks();
//drinks function is called
        break;
    default:
        printf("\n you pressed wrong number \n");
        goto again;
//if user gives wrong input
    }
    cash=deposit();
    purse=cashback(cash);
    again();
    return 0;
}
void welcome(){
//prints welcome message to the user
    for(int i=0;i ⟨30;i++){
       printf("*");
    }
    printf("\t \t \n welcome to vending machine \n");
    for (int i = 0; i ⟨ 30; i++)
    {
        printf("*");
    }
}
void rules(){ //prints rules
    printf("\n please don't enter coins in the machine.\n");
    printf("you can buy a single type of item at a time.\n");
    printf("In case if your money get stuck inside machine please contact to 0755-658138.\n");
    printf("please don't kick or stomp the machine.\n");
    for(int i = 0 ;i⟨30;i++){
        printf("*");
    }
}
void snacks(){
```

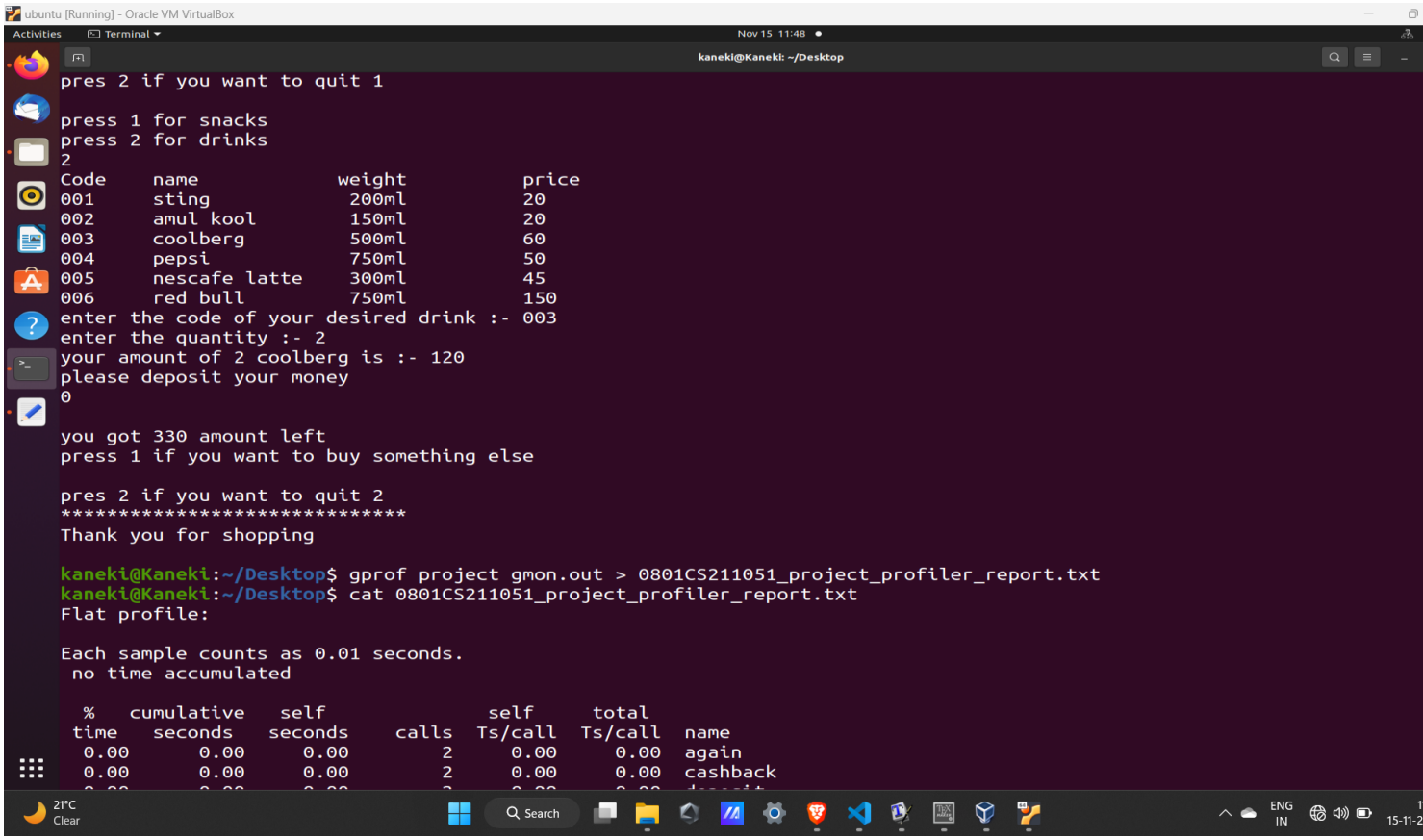
```
//tells what all snacks are available in vending machine
    printf("Code \t name \t \t weight \t \t price\n");
    printf("001\t kurkure \t \t 110gm \t \t 20\n");
    printf("002 \t lays \t \t 100gm \t \t 20\n");
    printf("003 \t sandwich \t 200gm \t \t 60 \n");
    printf("004\t soya sticks \t 150gm \t \t 50 \n");
    amount =selection_snacks();
//selection_snacks function is called and its returned value is stored in amount
}
void drinks(){
//tells what all drinks are available in vending machine
    printf("Code \t name \t \t weight \t \t price \n");
    printf("001 \t sting \t \t 200ml \t \t 20 \n");
    printf("002 \t amul kool \t 150ml \t \t 20 \n");
    printf("003 \t coolberg \t 500ml \t \t 60 \n");
    printf("004 \t pepsi \t \t 750ml \t \t 50 \n");
    printf("005 \t nescafe latte \t 300ml \t \t 45 \n");
    printf("006 \t red bull \t 750ml \t \t 150 \n");
    amount=selection_drinks();
//selection_drinks function is called and its returned value is stored in amount
}
int selection_snacks(){
//It ask for the code and quantity of the product and calculates total amount of the product
    int code,quantity,total_amount;
    again_selection:
    printf("enter the code of your desired snack :- ");
    scanf("%d",&code);
    printf("enter the quantity :- ");
    scanf("%d",&quantity);
    if (code == 001)
    {
        total_amount=20*quantity;
        printf("your amount of %d kurkure is :- %d",quantity,total_amount);
    }
    else if (code == 002)
    {
        total_amount=20*quantity;
        printf("your amount of %d lays is :- %d",quantity,total_amount);
    }
    else if (code == 003)
    {
        total_amount=45*quantity;
        printf("your amount of %d sandwich is :- %d",quantity,total_amount);
    }
    else if (code == 004)
    {
        total_amount=50*quantity;
        printf("your amount of %d soya sticks is :- %d",quantity,total_amount);
    }
    else{
        printf("\n you enter wrong code \n");
        goto again_selection;
//It ask for the code and quantity of the product and calculates total amount of the product
    }       return total_amount;
}
int selection_drinks(){
//It ask for the code and quantity of the product and calculates total amount of the product
    int code,quantity,total_amount;
    again_selection:
    printf("enter the code of your desired drink :- ");
    scanf("%d",&code);
    printf("enter the quantity :- ");
    scanf("%d",&quantity);       if (code == 001)
```

```
    {
        total_amount=20*quantity;
        printf("your amount of %d sting is :- %d",quantity,total_amount);
    }
    else if (code == 002)
    {
        total_amount=20*quantity;
        printf("your amount of %d amul kool is :- %d",quantity,total_amount);
    }
    else if (code == 003)
    {
        total_amount=60*quantity;
        printf("your amount of %d coolberg is :- %d",quantity,total_amount);
    }
    else if (code == 004)
    {
        total_amount=50*quantity;
        printf("your amount of %d pepsi is :-       }
    else if (code == 005)
    {
        total_amount=45*quantity;
        printf("your amount of %d nescafe latte is :- %d",quantity,total_amount);
    }
    else if (code == 006)
    {
        total_amount=150*quantity;
        printf("your amount of %d red bull is :- %d",quantity,total_amount);
    }
    else{
        printf("\n you enter wrong code \n");
        goto again_selection;
//It ask for the code and quantity of the product and calculates total amount of the product
    }
    return total_amount;
}
int deposit(){
//it takes deposit from the user and store it in cash variable
    int deposit;
    printf("\n please deposit your money \n");
    scanf("%d",&deposit);
    return deposit;
}
/*it calculates that is there any shortage of money in the deposited amount and tells the left amount in the purse. It returns remaining*-1 to purse variable in the main function. */
int cashback(int cash){
    int remaining;
    remaining = amount - cash;
    if(remaining⟩0){
        remaining=shortage_deposit(remaining,cash);
       }
    printf("\n you got %d amount left \n",remaining*-1);
    return remaining*-1;
}
/*it tells the user on how much money he is short on and takes the remaining amount and then also if the user is in shortage it will go in recursion until the amount = deposit.it returns the remaining to function cashback.*/
int shortage_deposit(int remaining,int cash){
    int shortage,left;
    remaining=cash - amount;
    printf("you are short on %d amount \n",remaining*-1);
    printf("please deposit the remaining amount \n");
    scanf("%d",&shortage);
    remaining = (remaining*-1)-shortage;
    cash = cash+shortage;
```

```
    if (remaining⟩0)
    {
        shortage_deposit(remaining,cash);
    }
    else{
        return remaining;
    }
}
/∗it asks the that does he want to buy again anything and if the user wants to buy something then shop_again
function will be called else thank you message will be printed.∗/
void again(){
    int response;
    try_again:
    printf("press 1 if you want to buy something else \n ");
    printf("\n pres 2 if you want to quit ");
    scanf("%d",&response);
    switch (response)
    {
    case 1:
         shop_again();
         break;
    case 2:
    for (int i = 0; i ⟨ 30; i++)
    {
         printf("∗");
    }
    printf("\n Thank you for shopping \n");
    printf("\n please visit us again \n");
    for (int i = 0; i ⟨ 30; i++)
    {
        printf("∗");
    }
    break;
    default:
        printf("you pressed wrong number");
       goto try_again;
        break;
    }
 }
/∗ it calls all the function again except welcome and rules function and add the last remaining amount in the purse
for future purchases ∗/
void shop_again(){
    int selection,cash;
    again:
    printf("\n press 1 for snacks \n press 2 for drinks \n");
    scanf("%d",&selection);
    switch (selection)
    {
    case 1:
         snacks();
         break;
    case 2:
         drinks();
         break;
    default:
         printf("\n you pressed wrong number \n");
         goto again;
    }
    cash=deposit()+purse;
    cashback(cash);
    again();
}
```

Output of the code





Code in Python Language

```
global amount
global purse
purse = 0
def welcome():
#prints welcome message to the user
    for i in range (0,30):
        print("*",end=")
    print(" \n welcome to vending machine \n")
    for i in range (0,30):
        print("*",end=")

def rules():
#prints rules
    print(" \n please don't enter coins in the machine.\n ")
    print("you can buy a single type of item at a time.\n ")
    print("In case if your money get stuck inside machine please contact to 0755-658138. \n ")
    print("please don't kick or stomp the machine. \t ")

def snack():
#tells what all snacks are available in vending machine
    print("Code\t name\t \t weight \t \t price \n ")
    print("101 \t kurkure \t \t 110gm \t \t 20 \n")
    print("102 \t lays \t \t 100gm \t \t 20 \n")
    print("103 \t sandwich \t 200gm \t \t 60 \n")
    print("104 \t soya sticks \t 150gm \t \t 50 \n")
    amount = selection_snacks()
#selection_snacks function is called and its returned value is stored in amount
    return amount

def selection_snacks():
#It ask for the code and quantity of the product and calculates total amount of the product
    total_amount=0
    code = int(input("enter the code of your desired snack \n "))
    quantity=int(input("enter the quantity \n"))
    if code == 101:
        total_amount = 20*quantity
        print("your amount for ",quantity,"kurkure is ",total_amount)
    elif code == 102:
        total_amount=20*quantity
        print("your amount for ",quantity,"lays is ",total_amount)
    elif code == 103:
        total_amount=60*quantity
        print("your amount for ",quantity,"sandwich is ",total_amount)
    elif code == 104:
        total_amount=50*quantity
        print("your amount for ",quantity,"soya sticks is ",total_amount)
    return total_amount

def drinks():
#tells what all drinks are available in vending machine
    print("Code \t name \t \t weight \t \t price\n")
    print("101 \t sting \t \t 200ml \t \t 20 \n")
    print("102 \t amul kool \t 150ml \t \t 20 \n")
    print("103 \t coolberg \t 500ml \t \t 60 \n")
    print("104 \t pepsi \t \t 750ml \t \t 50 \n")
    print("105 \t nescafe latte \t 300ml \t \t 45 \n")
    print("106 \t red bull \t 750ml \t \t 150 \n")
    amount=selection_drinks()
#selection_drinks function is called and its returned value is stored in amount
    return amount
```

```
def selection_drinks():
#It ask for the code and quantity of the product and calculates total amount of the product
    total_amount=0
    code = int(input(" \n enter the code of your desired drink \n"))
    quantity=int(input("\n enter the quantity \n"))
    if code == 101:
        total_amount = 20*quantity
        print("your amount for ",quantity,"sting is ",total_amount)
    elif code == 102:
        total_amount=20*quantity
        print("your amount for ",quantity,"amul kool is ",total_amount)
    elif code == 103:
        total_amount=60*quantity
        print("your amount for ",quantity,"coolberg is ",total_amount)
    elif code == 104:
        total_amount=50*quantity
        print("your amount for ",quantity,"pepsi is ",total_amount)
    elif code == 105:
        total_amount=45*quantity
        print("your amount for ",quantity,"nescafe latte is ",total_amount)
    elif code == 106:
        total_amount=150*quantity
        print("your amount for ",quantity,"red bull is ",total_amount)
    return total_amount

def deposit():
#it takes deposit from the user and store it in cash variable
    deposit = int(input("please deposit your money " ))
    return deposit

"""
It calculates that is there any shortage of money in the deposited amount and tells the left amount in the purse. It returns remaining*-1 to purse variable in the main function.
"""
def cashback(cash,amount):
    remaining:any
    left=0
    remaining=amount-cash
    if remaining⟩0:
        remaining=shortage_deposit(remaining,cash)
    left=remaining*-1
    print("\n you got ",left,"amount left in your purse")
    return left

"""
It tells the user on how much money he is short on and takes the remaining amount and then also if the user is in shortage it will go in recursion until the amount = deposit.it returns the remaining to function cashback.
"""
def shortage_deposit(remaining,cash):
    shortage:any
    purse
    remaining=cash-amount
    print("you are short on ",remaining*-1,"amount \n")
    shortage = int(input("please deposit the remaining amount \n"))
    remaining = (remaining*-1)-shortage
    cash=cash+shortage
    if remaining⟩0:
        shortage_deposit(remaining,cash)
    else:
        return remaining

"""
it asks the that does he want to buy again anything and if the user wants to buy something then shop_again function will be called else thank you message will be printed.
```

```
"""
def again(purse):
    response:any
    print("press 1 if you want to buy something else \n")
    print("press 2 if you want to quit")
    response=int(input())
    if response == 1:
        shop_again(purse)
    else:
        for i in range (0,30):
            print("*",end='')
        print("\n Thank you for shopping \n")
        print("\n Please visit us again \n")
        for i in range(0,30):
            print("*",end='')

"""
It calls all the function again except welcome and rules function and add the last remaining amount in the purse for future purchases
"""
def shop_again(purse):
    selection:any
    cash:any
    print("\n press 1 for snacks and press 2 for drinks \n")
    selection=int(input())
    if selection==1:
        amount=snack()
    else:
        amount=drinks()
    cash=deposit()+purse;
    purse=cashback(cash,amount);
    again(purse);
    return 0;

selection:any
cash:any
welcome()
rules()
print("\n press 1 for snacks and press 2 for drinks \n")
selection=int(input())
if selection==1:
#selection for whether user want snacks or drinks
    amount=snack()
#snacks function is called else:
    amount=drinks()
#drinks function is called
cash=deposit()
purse=cashback(cash,amount)
again(purse)
```

Output of Python code