CODE AND TESTING

GROUP - 16

190101028 - Bolleboina Madhumitha
190101030 - Chappidi Shreya
190101042 - Kalapati Kasvitha
190101056 - Mushanolla Pranathi

I. FUNCTION MAPPING

MODULE 1 - Authentication

Sub-module	Function
SignUp	int signup(string name, string email, string password)
LogIn	int login(string email, string password)
Forgot Password	int forgotpassword(string email, string oldpassword, string newpassword)
LogOut	int logout()

MODULE 2 - User Settings

Sub-module	Function
User Settings	int user_settings(string name, long long phone no, string email, string password)

MODULE 3 - Item

Sub-module	Function
New Item	void new_item(string name, int price)
Items List	vector<Item> items_list()
Edit Item	bool edit_item(string oldname, string newname, string newprice)
Delete Item	bool delete_item(string name)

MODULE 4 - Customer

Sub-module	Function
New Customer	void new_customer(string name, int price)
Customers List	vector<Customer> customers_list()
Edit Customer	bool edit_customer(string oldemail, long long newphoneno, string newemail)
Delete Customer	bool delete_customer(string email)

MODULE 5 - Invoice

Sub-module	Function
New Invoice	void new_invoice(Customer customer, Item item, int invoiceno, long long amount)
Invoices List	vector<Invoice> invoices_list()
Edit Invoice	bool edit_invoice(Customer oldcustomer, Item olditem, Customer newcustomer, Item newitem, long long newamount)
Delete Invoice	bool delete_invoice(Customer customer, Item item)
Send Invoice	int send_invoice()

MODULE 6 - Payment

Sub-module	Function
New Payment	void new_payment(Customer customer, Item item, int paymentno, long long amount)
Payments List	vector<Payment> payments_list()
Edit Payment	bool edit_payment(Customer oldcustomer, Item olditem, Customer newcustomer, Item newitem, long long newamount)
Delete Payment	bool delete_payment(Customer customer, Item item)
Send Payment	int send_payment()

MODULE 7 - Expense

Sub-module	Function
New Expense	void new_expense(string date, long long amount, string note)
Expenses List	vector<Expense> expenses_list()
Edit Expense	bool edit_expense(string olddate, string newdate, long long newamount, string newnote)
Delete Expense	bool delete_expense(string email)

MODULE 8 - Report

Sub-module	Function
Sales	File* sales(string startdate, string enddate, string category)
Expenses	File* expenses(string startdate, string enddate, string category)
Profits/Losses	File* profits_losses(string startdate, string enddate, string category)

MODULE 9 - Analysis

Sub-module	Function
Analysis	File* analysis()

II. CODE IMPLEMENTATION

*only 3 important functions are described here

Function 1 : NEW INVOICE

Description

New invoice is created when the accountant enters the details of customer , Item, amount, invoice number.

Line 3 - System checks if the customer emailid is existing in the database.

Line 4 - System gets the customer information.

Line 8 - System gets new customers name

Line 9 - System checks if the entered customer name is a valid name

Line 13 - System gets new customers phone number

Line 14 - System checks if the entered number is a valid number

Line 17 - System adds the new customer details in the database

Line 19 - System checks if the item is existing in database

Line 20 - System gets the item information

Line 24 - System gets the item price information

Line 28 - System adds the item into database

Line 36 - Systems adds the new invoice into the database

```
1   void new_invoice (Customer customer, Item item, int invoiceno, long long amount) {
2
3       if(search_customer(customer.email))
4               customer = customers[searched_customer_index];
5       else {
6               cout << "Its a new customer" << endl;
7               cout << "Enter customer name : ";
8               cin >> customername;
9               if(check(customername))
10                  customer.name = customername;
11              else return;
12              cout << "Enter phone number : ";
13              cin >> phoneno;
14              if(6*1000000000<=phoneno && phoneno<10000000000)
15                  customer.phoneno = phoneno;
16              else return;
17              new_customer(customername,phoneno,customer.email);
18          }
19      if(search_item(item.name))
20              item = items[searched_item_index];
21      else {
22              cout << "Its a new item" << endl;
23              cout << "Enter item price : ";
24              cin >> price;
25              if(0<=price && price<=1000000000)
26                  item.price = price;
27              else return;
28              new_item(item.name,price); item.company = company;
29          }
30      Invoice in;
31      in.customer = customer;
32      in.item = item;
33      in.invoiceno = invoiceno;
34      in.company = company;
35      in.amount = amount;
36      invoices.push_back(in);
37      cout << "Invoice added and sent succesfully" << endl;
38  }
```

Function 2 : EDIT INVOICE

Description

Invoice is edited when the accountant reenters the details of customer, Item, amount, invoice number.

Line 2 - System checks if the customer emailid is existing in the database.

Line 4 - System checks the new mail is valid.

Line 5 - System updates email

Line 7 - System checks if the item is in database

Line 8 - System updates the new item details

Line 10 - System checks the item price is valid or not

Line 11 - System updates the item price

```
bool edit_invoice (Customer customer1, Item item1, Customer customer2, Item item2, lon
    if(search_invoice(customer1.email,item1.name)) {
        int ind = searched_invoice_index;
        if(customer2.email.substr(customer2.email.length()-10,10)=="@gmail.com")
            invoices[ind].customer = customer2;
        else return false;
        if(check(item2.name))
            invoices[ind].item = item2;
        else return false;
        if(0<=amount2 && amount2<=1000000000)
            invoices[ind].amount = amount2;
        else return false;
        return true;
    }
    else return false;
}
```

Function 3 : DELETE INVOICE

Description

Invoice is deleted when the accountant enters the details of customer, Item.

Line 2 - System checks if the customer emailid is existing in the database.

Line 5 - System deletes the invoice

```
bool delete_invoice (Customer customer, Item item) {
    if(search_invoice(customer.email,item.name))
    {
        int ind = searched_invoice_index;
        invoices.erase(invoices.begin()+ind);
        return true;
    }
    else return false;
}
```

III. BLACK BOX TESTING

FUNCTION 1 : NEW INVOICE

The test cases for the New Invoice were derived by **Equivalence Class Analysis including boundary values**. The cases for valid input were chosen so that for each variable, one value from each invalid equivalence class was selected. So that in each case, only one of the variables had an invalid value and the rest were valid. This resulted in 15 test cases. 7 of them were for valid inputs and others were for invalid inputs.

The following assumptions were made:
The customer email variable has its suffix as "@gmail.com"
The phone number variable has a maximum value of 9999999999 and minimum value of 6000000000.
The company name ,customer name should be only in alphabets also size >=2 && <=30
The amount and invoice number is maximum of 1000000000
The equivalence classes are as follows:

Variable	Equivalence class			
Customer email	Suffix is "@gmail.com"	Suffix is not "@gmail.com"		
Customer phone	<6000000000	[6000000000-9999999999]	>9999999999	
Customer Name	String have characters which are not alphabets	String is made of only alphabets and the string size is <2	String is made of only alphabets and the string size is [2,30]	String is made of only alphabets and the string size is >30
Company name	String have characters which are not alphabets	String is made of only alphabets and the string size is <2	String is made of only alphabets and the string size is [2,30]	String is made of only alphabets and the string size is >30
Amount	[0-1000000000]	>10000000000	<0	
Invoice number	[0-1000000000]	>1000000000	<0	
Item name	the string size is >30	the string size is <2	the string size is [2,30]	

The variable values used for the test cases include:

1. Customer email
   a. Suffix is "@gmail.com" → pranathi@gmail.com
   b. Suffix is not "@gmail.com" → pranathi.com
2. Customer phone
   a. <6000000000 → 145678
   b. [6000000000-9999999999] → 9999988888
   c. >9999999999 → 100000000000
3. Customer name
   a. String have characters which are not alphabets → pranathi%%12
   b. String have characters which are alphabets and
      The string size is less than 2 → m
   c. String have characters which are alphabets and
      The string size is [2,30] → pranathi
   d. String have characters which are alphabets and
      The string size is >40 → pranathiaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
4. Customer name
   a. String have characters which are not alphabets → Google%23
   b. String have characters which are alphabets and
      The string size is less than 2 → G
   c. String have characters which are alphabets and
      The string size is [2,40] → Google
   d. String have characters which are alphabets and
      The string size is >40 → Googleaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
5. Item name
   a. The string size is less than 2 → G
   b. The string size is [2,40] → calendar
   c. The string size is >40 → calendaraaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
6. amount
   a. [0-1000000000] → 12345678
   b. >1000000000 → 10000000000000000000000
   c. <0 → -1
7. Invoice number
   a. [0-1000000000] → 12345678
   b. >1000000000 → 10000000000000000000000
   c. <0 → -1

Test Cases

Each test case is taken from each one of the **Equivalence classes** and the **boundary values** of all the classes is taken as testcase

Test case	Email	Phone	Name	Company	Amount	Invoice num	Item name
1	Pranathi@gmail.com	9999988888	pranathi	Google	0	190101056	Calendar
2	pranav@gmail.com	9999988888	Shreya	Google	1000000000	190101042	Calendar1
3	shreya@gmail.com	9999988888	pranathi	Google	1000000001	190101057	Calendar2
4	kasvitha@gmail.com	6999999999	pranathi	microsoft	99999999	190101058	Calendar3
5	madhu@gmail.com	6000000000	manvi	Microsoft	1000000	190101059	teams
6	Pranathi@gmail.com	6000000001	pranathi%%12	Github	50000	190101060	I
7	Pranathi@gmail.com	5999999999	Pranathiaaaaa Aaaaaaaaaaa aaaaaaaa aaaaaaaa	Linkedin	-1	190101061	Calendar Aaaaaaaaa Aaaaaaaaaa aaaaaaaaaaaa
8	Pranathi.com	10000000000	shreya	Facebook	100000000000	190101062	Item1
9	Pranathi@gmail.com	145678	madhu	Google%23	80000	190101066	Item2
10	Pranathi@gmail.com	9999988888	Kasvitha	G	90000	190101067	Item3
11	Pranathi@gmail.com	9999988888	Pranathi	Googleaaa Aaaaaaaaaaa aa aaaaaaaaaa aaaaaa	10000	100000000000	Item4
12	Pranathi@gmail.com	9999988888	kasvitha	Minium	100000	0	Item5
13	Pranathi@gmail.com	9999988888	Shreya	facebook	200000	1000000000	Item6
14	Pranathi@gmail.com	9999988888	Pranathi	Whatsapp	300000	1000000001	Item7
15	Pranathi@gmail.com	9999988888	kasvitha	amazon	900000	-1	Item8

Results

Testcase	Result
1	Success
2	Success
3	Success
4	Success
5	Success
6	Fail
7	Fail
8	Fail
9	Fail
10	Fail
11	Fail
12	Fail
13	Success
14	Success
15	Fail
16	Fail

FUNCTION 2 - EDIT INVOICE

The test cases for the Edit Invoice were derived by **Equivalence Class Analysis including boundary values**. The cases for valid input were chosen so that for each variable, one value from each invalid equivalence class was selected. So that in each case, only one of the variables had an invalid value and the rest were valid. This resulted in 15 test cases. 8 of them were for valid inputs and others were for invalid inputs.

The following assumptions were made:
The customer email variable has its suffix as “@gmail.com”
The company name should be only in alphabets also size >=2 && <=30
The amount and invoice number is maximum of 1000000000
The equivalence classes are as follows:

Variable	Equivalence class			
Customer email	Suffix is “@gmail.com”	Suffix is not “@gmail.com”		
Company name	String have characters which are not alphabets	String is made of only alphabets and the string size is <2	String is made of only alphabets and the string size is [2,50]	String is made of only alphabets and the string size is >50
Amount	[0-1000000000]	>10000000000	<0	
Invoice number	[0-1000000000]	>1000000000	<0	
Item name	the string size is >50	the string size is <2	the string size is [2,50]	

The variable values used for the test cases include :

1. Customer email
   a. Suffix is “@gmail.com” → pranathi@gmail.com
   b. Suffix is not “@gmail.com” → pranathi.com
2. Customer name
   a. String have characters which are not alphabets -> Google%23
   b. String have characters which are alphabets and
   The string size is less than 2 → G
   c. String have characters which are alphabets and
   The string size is [2,40] → Google
   d. String have characters which are alphabets and
   The string size is >40 → Googleaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
3. amount
   a. [0-1000000000] → 12345678
   b. >1000000000 → 10000000000000000000000
   c. <0 → -1
4. Invoice number
   a. [0-1000000000] → 12345678
   b. >1000000000 → 10000000000000000000000
   c. <0 → -1
5. Item name
   a. The string size is less than 2 → G
   b. The string size is [2,40] → calendar
   c. The string size is >40 → calendaraaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Test Cases

Each test case is taken from each one of the Equivalence classes and the boundary values of all the classes is taken as testcase

Test case	Email	Company	Amount	Invoice num	Item name
1	Pranathi@gmail.com	Google	0	190101056	calendar
2	pranav@gmail.com	Google	1000000000	190101042	Calendar1
3	shreya@gmail.com	Google	1000000001	190101057	Calendar2
4	kasvitha@gmail.com	microsoft	99999999	190101058	Calendar3
5	madhu@gmail.com	Microsoft	1000000	190101059	teams
6	Pranathi@gmail.com	Github	50000	190101060	I
7	Pranathi@gmail.com	Linkedin	-1	190101061	Calendar Aaaaaaaaa Aaaaaaaaaa aaaaaaaaaaaa
8	Pranathi.com	Facebook	100000000000	190101062	Item1
9	Pranathi@gmail.com	Google%23	80000	190101066	Item2
10	Pranathi@gmail.com	G	90000	190101067	Item3
11	Pranathi@gmail.com	Googleaaa Aaaaaaaaaaaa aa aaaaaaaaaa aaaaaa	10000	100000000000	Item4
12	Pranathi@gmail.com	Minium	100000	0	Item5
13	Pranathi@gmail.com	facebook	200000	1000000000	Item6
14	Pranathi@gmail.com	Whatsapp	300000	1000000001	Item7
15	Pranathi@gmail.com	amazon	900000	-1	Item8

Results

Testcase	Result
1	Success
2	Success
3	Success
4	Success
5	Success
6	Success
7	Fail
8	Fail
9	Fail
10	Fail
11	Fail
12	Fail
13	Success
14	Success
15	Fail

FUNCTION 3 - DELETE INVOICE

The test cases for the Delete Invoice were derived by **Equivalence Class Analysis including boundary values**. The cases for valid input were chosen so that for each variable, one value from each invalid equivalence class was selected. So that in each case, only one of the variables had an invalid value and the rest were valid. This resulted in 8 test cases. 5 of them were for valid inputs and others were for invalid inputs.

The following assumptions were made:
The customer email variable has its suffix as “@gmail.com”
The item name size >=2 && <=30
The equivalence classes are as follows:

Variable	Equivalence class		
Customer email	Suffix is “@gmail.com”	Suffix is not “@gmail.com”	
Item name	the string size is >50	the string size is <2	the string size is [2,50]

The variable values used for the test cases include:

1. Customer email
   a. Suffix is “@gmail.com” → pranathi@gmail.com
   b. Suffix is not “@gmail.com” → pranathi.com
2. Item name
   a. The string size is less than 2 → G
   b. The string size is [2,40] → calendar
   c. The string size is >40 → calendaraaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Test Cases

Each test case is taken from each one of the Equivalence classes and the boundary values of all the classes is taken as testcase

Testcase	Email	Item name
1	Pranathi.com	calendar
2	pranav@gmail.com	Calendar1
3	shreya@gmail.com	Calendar2
4	kasvitha@gmail.com	Calendar3
5	madhu@gmail.com	teams
6	Pranathi@gmail.com	I
7	Pranathi@gmail.com	Calendar Aaaaaaaaa Aaaaaaaaaa aaaaaaaaaaaa
8	Pranathi.com	Item1

Results

Testcase	Result
1	Fail
2	Success
3	Success
4	Success
5	Success
6	Fail
7	Fail
8	Success

IV. WHITE BOX TESTING

DataBase-

The following database is considered in picture while designing the testcases for white box testing

Item name	Customer email
Item1	charitha@gmail.com
Item2	shravya@gmail.com

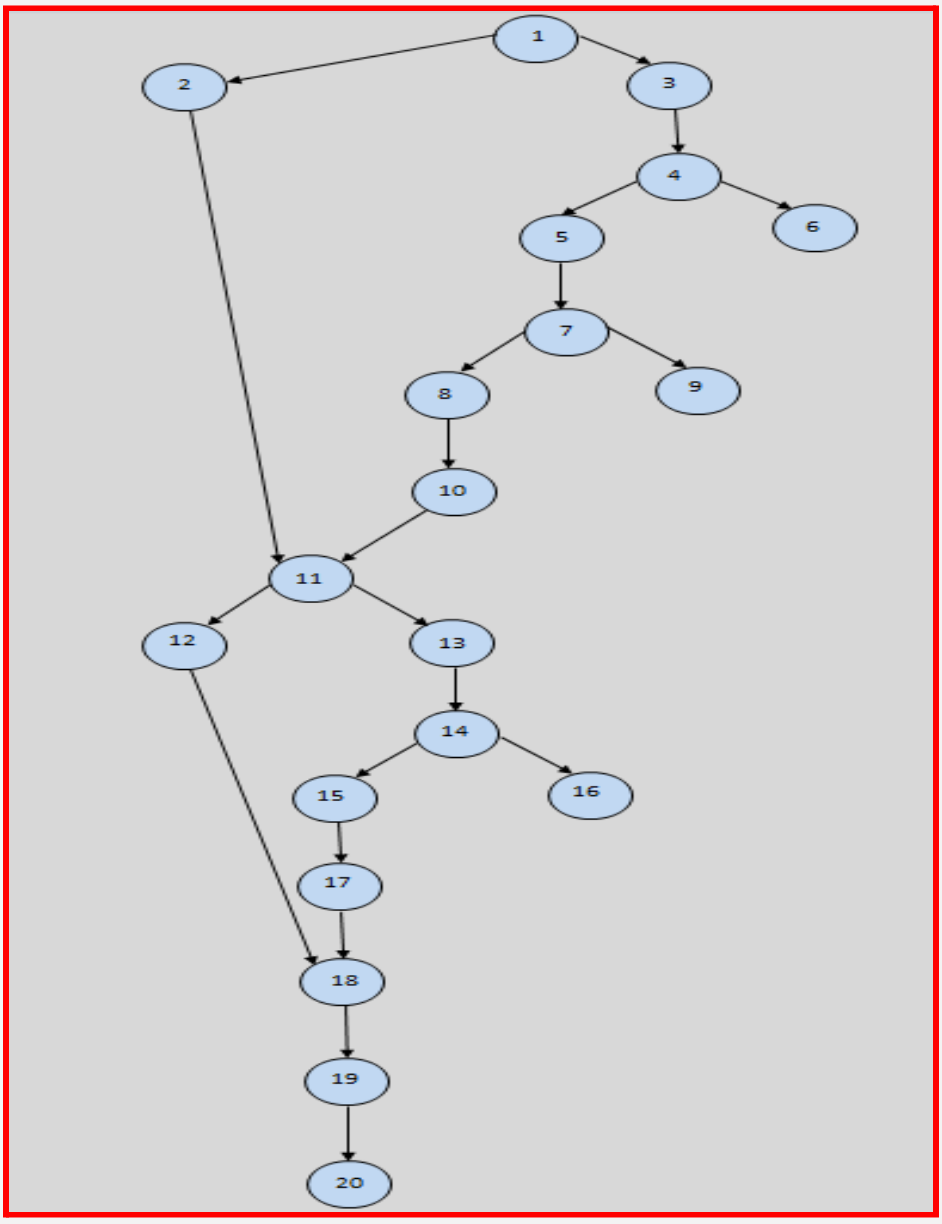
FUNCTION 1 - NEW INVOICE

The test cases for the New Invoice ensure all linearly independent paths in the code.
The cases are written which covers all the paths.

The following assumptions were made:
The customer email variable has its suffix as “@gmail.com”
The phone number variable has a maximum value of 9999999999 and minimum value of 6000000000.
The company name ,customer name should be only in alphabets also size >=2 && <=30
The amount and invoice number is maximum of 1000000000
The equivalence classes are as follows:

```
if(search_customer(customer.email))
    customer = customers[searched_customer_index];
else {
    if(check(customername))
        customer.name = customername;
    else return;
    if(6*1000000000<=phoneno && phoneno<10000000000)
        customer.phoneno = phoneno;
    else return;
    new_customer(customername,phoneno,customer.email); }
if(search_item(item.name))
    item = items[searched_item_index];
else {
    if(0<=price && price<=1000000000)
        item.price = price;
    else return;
    new_item(item.name,price); item.company = company; }
Invoice in;
in.customer = customer; in.item = item; in.invoiceno = invoiceno; in.company = company; in.amount = amount;
invoices.push_back(in);
```

CFG :



Testcase	Independent paths
1	1→3→4→6
2	1→2→3→4→5→7→9
3	1→2→3→4→5→7→8→10→11→13→14→16
4	1→3→4→5→7→8→10→11→13→14→16→17→18→19→20
5	1→3→4→5→7→8→10→11→12→18→19→20
6	1→2→11→13→14→16
7	1→2→11→13→14→15→17→18→19→20
8	1→2→11→12→18→19→20

Test Cases

Test case	Email	Name	Phone	Item name	Amount	Invoice num
1	Pranathi@gmail.com	Pranathi1923	9999999999	Item1	10,000	190101042
2	pranathi@gmail.com	pranathi	99999	Item1	10,000	190101042
3	pranathi@gmail.com	pranathi	9999999999	Item3	-1	190101057
4	pranathi@gmail.com	pranathi	9999999999	Item3	10,000	190101058
5	pranathi@gmail.com	pranathi	9999999999	Item1	10,000	190101058
6	charitha@gmail.com	charitha	9999999999	Item3	-1	190101057
7	charitha@gmail.com	charitha	9999999999	Item3	10,000	190101058
8	charitha@gmail.com	charitha	9999999999	Item1	10,000	190101058

Results

Testcase	Result
1	Fail
2	Fail
3	Fail
4	Success
5	Success
6	Fail
7	Success
8	Success

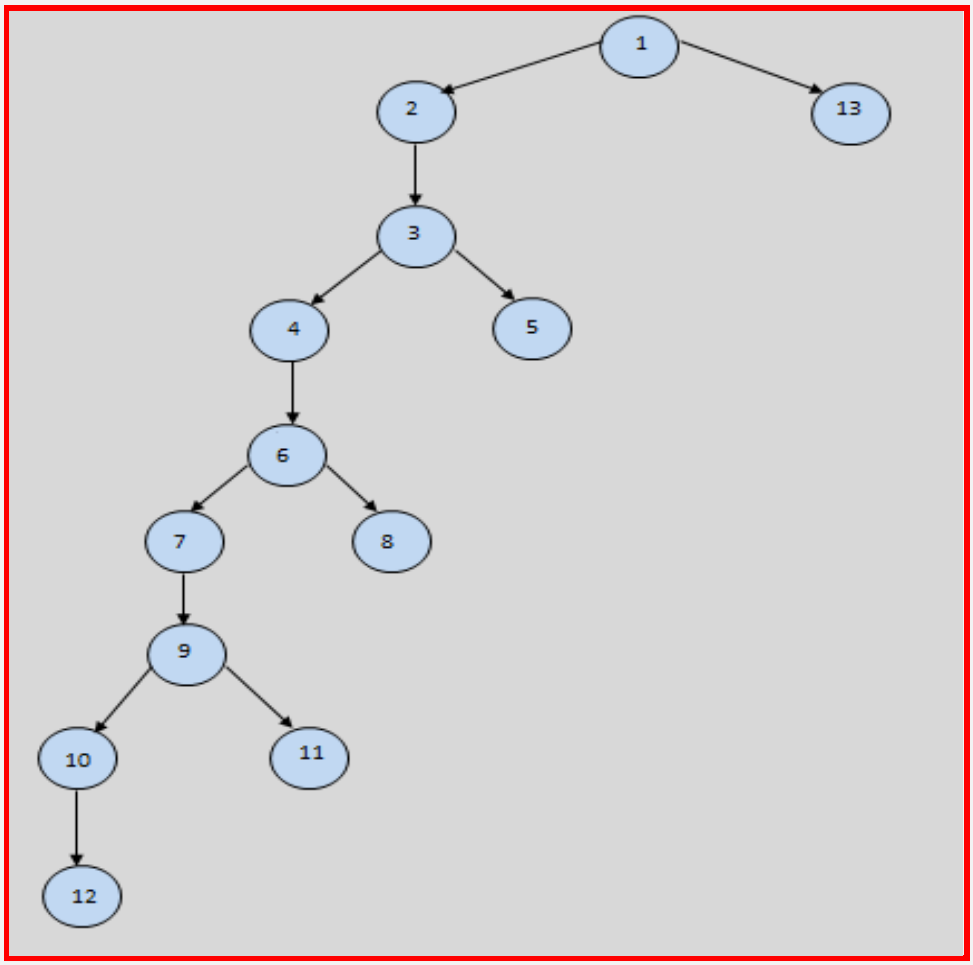
FUNCTION 2 - EDIT INVOICE

The test cases for the Edit Invoice ensure all linearly independent paths in the code.
The cases are written which covers all the paths.

The following assumptions were made:
The customer email variable has its suffix as “@gmail.com”
The company name should be only in alphabets also size >=2 && <=30
The amount and invoice number is maximum of 1000000000
The equivalence classes are as follows:

```
if(search_invoice(customer1.email,item1.name)) {
    int ind = searched_invoice_index;
    if(customer2.email.substr(customer2.email.length()-10,10)=="@gmail.com")
        invoices[ind].customer = customer2;
    else return false;
    if(check(item2.name))
        invoices[ind].item = item2;
    else return false;
    if(0<=amount2 && amount2<=1000000000)
        invoices[ind].amount = amount2;
    else return false;
    return true; }
else return false;
```

CFG :



Testcase	Independent paths
1	1→13
2	1→2→3→5
3	1→2→3→4→6→8
4	1→2→3→4→6→7→9→11
5	1→2→3→4→6→7→9→10→12

Test Cases

Test case	Email	Item name	Amount	Invoice num
1	Pranathi@gmail.com	Item1	10,000	190101056

2	Pranathi.com	Item1	10,000	190101042
3	pranathi@gmail.com	Item%	10,000	190101057
4	pranathi	Item1	-1	190101058
5	pranathi@gmail.com	Item1	10,000	190101059

Result

Testcase	Result
1	Success
2	Fail
3	Fail
4	Fail
5	Success

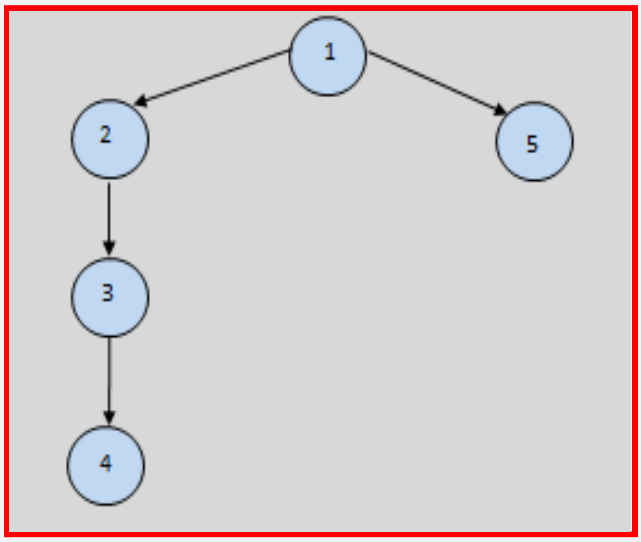
FUNCTION 3 - DELETE INVOICE

The test cases for the Delete Invoice ensure all linearly independent paths in the code. The cases are written which covers all the paths.

The following assumptions were made:
The customer email variable has its suffix as "@gmail.com"
The item name size >=2 && <=30
The equivalence classes are as follows:

```
bool delete_invoice (Customer customer, Item item) {
    if(search_invoice(customer.email,item.name)) {
        int ind = searched_invoice_index;
        invoices.erase(invoices.begin()+ind);
        return true;
    }
    else return false;
}
```

CFG :



Testcase	Independent paths
1	1→5
2	1→2→3→4

Test Cases

Testcase	Customer email	Item name
1	Pranathi@gmail.com	Item1
2	pranathi@gmail.com	Item3

Results

Testcase	Result
1	Success
2	Fail