INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT108-3-1-PYP

PYTHON PROGRAMMING

APD1F2109CS(IS), APU1F2109CS(IS), APD1F2109CS, APU1F2109CS(DF), APU1F2109CS, APD1F2109CS(DF), APD1F2109CS(CYB), APU1F2109CS(CYB)

HAND OUT DATE: 21st OCTOBER 2021

HAND IN DATE: 02nd DECEMBER 2021

WEIGHTAGE: 100%

STUDENT NAME: Mikael Owen

TP NUMBER: TP062732

INSTRUCTIONS TO CANDIDATES:

1. Submit your assignment online in Moodle Folder unless advised otherwise
2. Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

TABLE OF CONTENTS

Table of Contents

INTRODUCTION

Banking system has always been needed from over the years and has become people primary needs for years too. User experience according to banking system will impact much to how much money they put in that bank, Since then making the most efficient banking system is our main purpose. Again, by using our banking system we want to make sure that user is comfortable and the more efficient the more user will not be lazy to do transactions.

ASSUMPTION

1. I assume that there will be only 1 account provided by the system (Super User Account)
2. I assume that there will be only 1 login page
3. I assume that all user data is in data.txt
4. I assume that admin can only be generated by super user account
5. I assume that customer's account can only be generated by admin account
6. I assume that all account usernames and passwords are generated in a sequence
7. I assume that ID will have ROLE+UNIQUENUMBER for example SUPER1000
8. I assume that name cannot be modified
9. I assume that only admin and the customers itself that can modify their personal data
10. I assume that customer will have to choose either savings account or current account
11. I assume that saving account starting balance is RM10 and current balance is RM500
12. I assume that every transaction history will be noted in report.txt
13. I assume that customer can print their transaction history in specified date
14. I assume that customer can peform transaction included deposit,withdrawal,and transfer.

Design of Program – Pseudocode

LOGIN MENU

```
DEFINE FUNCTION main()
DECLARE IDchecker,rec,accNumber,lineNumber,userData as string
    OUTPUT("==========Welcome to BlueBank!==========")
    LOOP rec in f
    WITH OPEN "data.txt" with read mode as f
        IDchecker.append(rec.split(":")[1])
    NEXT rec
    ENDLOOP
     accNumber = id_input(IDchecker)
     lineNumber = IDchecker.index(accNumber)
     userData = [rec.split(':')
     LOOP rec IN
     WITH open("data.txt") read mode as f
     f =f.readlines()[linenumber]
    pass_input(userData)
    NEXT rec
    account_type(userData, lineNumber)
ENDDEFINE

DEFINE FUNCTION id_input(IDchecker)
DECLARE user_ID as string
     user_ID = input("Enter your ID ")
    DOWHILE TRUE
        IF user_ID not IN IDchecker THEN
            OUTPUT("The ID you entered is wrong")
             DISPLAY("Enter your ID: ")
            READ user_
        ELSE
           break
        ENDIF
    RETURN user_ID
ENDDEFINE
```

```
DEFINE FUNCTION pass_input(userData)
     DISPLAY"Enter your password ")
     READ read_pass
     check_password = userData[2]
    OUTPUT(userData[2])
    DOWHILE TRUE
        IF read_pass != check_password THEN
            OUTPUT("The password you entered is wrong ")
             DISPLAY"Enter your password ")
             READ read_pass
        ELSE
            break
        ENDIF
ENDDEFINE

DEFINE FUNCTION account_type(userData, lineNumber)
     check_type = userData[0]
    IF check_type EQUALS "S"THEN
        super_menu(userData, lineNumber)
    ENDIF
    IF check_type EQUALS "A"THEN
        admin_menu(userData, lineNumber)
    ENDIF
    IF check_type EQUALS "CC" or check_type EQUALS "CS"THEN
        customer_menu(userData, lineNumber)
    ENDIF
ENDDEFINE
```

SUPER MENU

```
DEFINE FUNCTION super_menu(userData, lineNumber)

    OUTPUT("====================Welcome to Super User Menu====================")

    OUTPUT(" 1. Create an admin account 2. Change password  3. Log Out ")

     DISPLAY(" Enter your option 1/2/3 ")
     READ super_resp
    DOWHILE TRUE
        IF super_resp != "1" and super_resp != "2"THEN

            OUTPUT(" The response you entered is invalid ... ")

             DISPLAY(" Enter your option 1/2/3 ")
             READ super_resp
        ENDIF
        IF super_resp EQUALS "1"THEN
        userData_create("ADMIN", "A", super_menu, userData, lineNumber)
        ENDIF
        IF super_resp EQUALS "2"THEN
            change_password(userData, lineNumber)
        ENDIF
        IF super_resp EQUALS "3"THEN
            OUTPUT("       You have logged      out of your userData.   Thanks for using BlueBank!")
            main()
        ENDIF
ENDDEFINE
```

NEW USER REGISTRATION

```
DEFINE FUNCTION userData_create(x, acc_type, menu, userData, lineNumber)

    OUTPUT("  CREATE A NEW " + x + " ACCOUNT")

    OUTPUT("=" * 20)

     DISPLAY(" Enter Account owner's name: ")
     READ name
    input_check(name, 3)

     DISPLAY(" Enter Account owner's phone number: ")
     READ number
    input_check(number, 4)

     DISPLAY(" Enter Account owner's email address: ")
     READ email
    input_check(email, 5)

    create_id(name, number, email, acc_type, x, menu, userData, lineNumber)

    menu()
ENDDEFINE
```

CHANGE USER PASSWORD

```
DEFINE FUNCTION change_password(userData, lineNumber)THEN
    DISPLAY(" Enter your current password: ")
    READ read_pass
     check_pass = userData[1]
    DOWHILE TRUE

        IF read_pass != check_passTHEN

            OUTPUT(" The password you entered is invalid ... ")

             DISPLAY(" Enter your current password: ")
             READ read_pass
        ELSE
            break
        ENDIF
     DISPLAY" Enter a new password ")
     READ new_pass
    DOWHILE TRUE
        IF new_pass EQUALS check_pass
            OUTPUT(" Your new password should not be the same as your old password ")
             DISPLAY(" Enter a new password ")
             READ new_pass
        ELSE
            break
        ENDIF
    with open("data.txt", "r") as data
         data_content = data.readlines()
         data_content[lineNumber] = userData[0] + ":" + new_pass + ":" + userData[2] + ":" + userData[3] + ":" +
                                    userData[4] + ":" + userData[5] + ":" + userData[6] + ":"
    with open("data.txt", "w") as data
        data.writelines(data_content)
    OUTPUT(" Your password has been changed      Please login using      your new password")
    main()
ENDDEFINE
```

ADMIN MENU

```
DEFINE FUNCTION admin_menu(userData, lineNumber)
    OUTPUT("====================Welcome to Admin Staff Menu====================")
    OUTPUT(" 1. Create a new customer account 2. Edit customer details ")
    OUTPUT(" 3. See customer's statement of account report 4. Change password 5. Log Out")
     DISPLAY(" Select an activity 1/2/3/4/5")
     READ admin_response
    DOWHILE TRUE
        IF admin_response != "1" and admin_response != "2" and admin_response != "3" and admin_response != "4" and
        admin_response != "5"THEN
            OUTPUT(" The response you entered is invalid  ")
             DISPLAY(" Select an activity 1/2/3/4/5")
             READ admin_response
        ENDIF
        IF admin_response EQUALS "1"THEN
            userData_create("customer", customer_type(), admin_menu, userData, lineNumber)
        ENDIF
        IF admin_response EQUALS "2"THEN
            customer_edit(userData, lineNumber)
        ENDIF
        IF admin_response EQUALS "3"THEN
            customer_statement()
        ENDIF
        IF admin_response EQUALS "4"THEN
            change_password(userData, lineNumber)
        ENDIF
        IF admin_response EQUALS "5"THEN
            OUTPUT("       You have logged      out of your userData.   Thanks for using BlueBank!")
            main()
        ENDIF
ENDDEFINE
```

EDIT CUSTOMER DATA

```
DEFINE FUNCTION customer_edit(userData, admin_line)
     DISPLAY(" Enter the account number you wish to modify ")
     READ id
     id_check = [x.split('')[1] LOOP x IN open("data.txt").readlines()]
    NEXT x
     lineNumber = id_check.index(id)
     customer_userData = [x.split(':') LOOP x IN open("data.txt").readlines()][lineNumber]
    NEXT x
    DOWHILE TRUE
        IF id not IN id_checkTHEN
            OUTPUT(" The account number you entered is invalid ")
            DISPLAY(" Enter the account number you wish to modify ")
            READ id
        ELSE
            break
        ENDIF
    customer_details(customer_userData)
     DISPLAY(" Select an information you would like = change (1/2): ")
     READ change_info
    DOWHILE TRUE
        IF change_info != "1" and change_info != "2"THEN
            OUTPUT(" The response you entered is invalid ... ")
              DISPLAY(" Select an information you would like = change (1/2): ")
              READ change_info
        ENDIF
        IF change_info EQUALS "1"THEN
             DISPLAY(" Enter customer's new phone number: ")
             READ new_number
            DOWHILE TRUE
        ENDIF
                IF new_number EQUALS customer_userData[4]THEN
OUTPUT(" The new number cannot be the same to the old one ")
DISPLAY(" Enter customer's new phone number: ")READ new_number

                ELSE
```

```
        OPEN("data.txt")   with read mode as data
    data_content = data.readlines()
 data_content[lineNumber] = customer_userData[0] + ":" + customer_userData[1] + ":" + \
                            customer_userData[2] + ":" + customer_userData[
                                3] + ":" + new_number + ":" + customer_userData[5] + ":" + \
                            customer_userData[6] + ":"
OPEN("data.txt")   with write mode as data
    data.writelines(data_content)
OUTPUT(" Current phone number: " + new_number + "")
admin_menu(userData, admin_line)

                ENDIF
        IF change_info EQUALS "2"THEN

             DISPLAY(" Enter customer's new email address: ")
             READ new_email
        ENDIF
            DOWHILE TRUE
                IF new_email EQUALS customer_userData[5]
OUTPUT(" The new email cannot be the same to the old one ")
DISPLAY(" Enter customer's new email address: ")READ new_email
                ELSE
OPEN("data.txt")   with read mode as data
     data_content = data.readlines()
     data_content[lineNumber] = customer_userData[0] + ":" + customer_userData[1] + ":" + \
                                customer_userData[2] + ":" + customer_userData[3] + ":" + \
                                customer_userData[4] + ":" + new_email + ":" + customer_userData[
                                    6] +
OPEN("data.txt")   with write mode as data
    data.writelines(data_content)
OUTPUT(" Current email address: " + new_email + "")
admin_menu(userData, admin_line)
     ENDIF
          ENDWHILE
ENDDEFINE
```

CHECK USER BALANCE

```
DEFINE FUNCTION user_balance(userData, lineNumber)
DECLARE balance as integer
    balance = userData[6]
    DISPLAY(" Type 1 = check user balance 2 = check record IN specific date 3 = back ")
    READ resp
    DOWHILE TRUE
        IF resp EQUALS "1" THEN
            OUTPUT("Your current balance is" + balance)
        ENDIF
        IF resp EQUALS "2" THEN
            dateRange()
        ENDIF
        IF resp EQUALS "3" THEN
            customer_menu(userData, lineNumber)
        ELSE
            OUTPUT("The option you entered is invalidd")
        ENDIF
ENDDEFINE
```

PREVENT SAME ACCOUNT DATA

```
DEFINE FUNCTION input_check(input_item, index_num)
     check_input = [rec.split(':')[index_num] LOOP rec IN open("data.txt").readlines()]
    IF index_num EQUALS 3THEN
        input_type = "name"
    ENDIF
    IF index_num EQUALS 4THEN
        input_type = "phone number"
    ENDIF
    IF index_num EQUALS 5THEN
        input_type = "email address"
    ENDIF
    DOWHILE TRUE
        IF input_item IN check_inputTHEN
            OUTPUT(" This " + input_type + " has been inputted ")
             DISPLAY(" Enter Account owner's " + input_type + ": ")
             READ input_item
        ELSE
            break
        ENDIF
    ENDWHILE
ENDDEFINE
```

CREATE NEW ACCOUNT and AUTO GENERATE ID AND PASSWORD

```
DEFINE FUNCTION create_id(name, number, email, acc_type, rec, menu, userData, lineNumber)
     existing_id = [rec.split(':')[1] LOOP rec IN open("data.txt").readlines()]
     new_pass = str(int(existing_id[-1][-4:]) + 1)
    IF acc_type EQUALS "CS" THEN
         new_id = "SAVINGS" + new_pass
         registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ":" + "100" + ""
    ENDIF
    IF acc_type EQUALS "CC" THEN
         new_id = "CURRENT" + new_pass
         registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ":" + "500" + ""
    ELSE
         new_id = "ADMIN" + new_pass
         registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ""
    ENDIF
    with open("data.txt", "a") as data :
    OPEN ("data.txt") with append mode as DATA
        data.write(registerdata)
     OUTPUT(" Account has been created ID = " + new_id + " Password = " + new_pass)
    OUTPUT(" Name : " + name + " Phone Number : " + number + " Email Address : " + email)
    menu(userData, lineNumber)
ENDDEFINE
DEFINE FUNCTION userData_create(x, acc_type, menu, userData, lineNumber)
    OUTPUT("  CREATE A NEW " + x + " ACCOUNT")
    OUTPUT("=" * 20)
     DISPLAY(" Enter Account owner's name: ")
     READ name
    input_check(name, 3)
     DISPLAY(" Enter Account owner's phone number: ")
     READ number
    input_check(number, 4)
     DISPLAY(" Enter Account owner's email address: ")
     READ email
    input_check(email, 5)
    create_id(name, number, email, acc_type, x, menu, userData, lineNumber)
ENDDEFINE
```

CHOOSING CUSTOMER TYPE AND SHOW DETAILS

```
DEFINE FUNCTION customer_type()
    OUTPUT(" Which customer account are you planning = create?")
     DISPLAY(" Savings account (CS) or Current account(CC): ")
     READ cust_type
    DOWHILE TRUE
        IF cust_type != "CS" and cust_type != "CC"THEN
            OUTPUT(" The response you entered is invalid ... ")
             DISPLAY(" Savings account (CS) or Current account(CC): ")
             READ cust_type
        ENDIF
        IF cust_type EQUALS "CS"THEN
             acc_type = "CS"
            break
        ENDIF
        IF cust_type EQUALS "CC"THEN

             acc_type = "CC"

            break
        ENDIF
    RETURN acc_type
ENDDEFINE

DEFINE FUNCTION customer_details(userData)
     customer_id, name, number, email = userData[1], userData[3], userData[4], userData[5]
    OUTPUT(" 1. ID :" + customer_id + " 2. Name : " + name + " 3. Phone Number : " + number + " 4. Email Address : " + email)
ENDDEFINE
```

TO PERFORM TRANSACTION (WITHDRAW)

```
DEFINE FUNCTION customer_withdraw(userData, lineNumber)
DECLARE balance as integer, acc_type as string
     balance = int(userData[6])
    acc_type = userData[0]
    DECLARE withdraw_amt as integer
    DISPLAY(" How much money you would like to withdraw "))
    READ withdraw_amt
    new_balance = balance - withdraw_amt
    DOWHILE TRUE
        IF balance <= withdraw_amt
            OUTPUT(" Your balance is not enough = withdraw  ")

            DECLARE withdraw_amt as integer
            DISPLAY(" How much money you would like to withdraw "))
            READ withdraw_amt
        ENDIF
        IF acc_type EQUALS "CS" and balance - withdraw_amt <= 100 THEN
            OUTPUT(" Your balance is not enough = withdraw  ")
            DECLARE withdraw_amt as integer
            DISPLAY(" How much money you would like to withdraw "))
            READ withdraw_amt
            break
        ENDIF
        IF acc_type EQUALS "CC" and balance - withdraw_amt <= 500 THEN
            OUTPUT(" Your balance is not enough = withdraw  ")
            DECLARE withdraw_amt as integer
            DISPLAY(" How much money you would like to withdraw "))
            READ withdraw_amt
            break
        ELSE
             new_balance = balance - withdraw_amt
        ENDIF
```

```
        with open("data.txt", "r") as data:
             data_content = data.readlines()
             data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + userData[2] + ":" + userData[3] + ":" +
                  userData[4] + ":" + userData[5] + ":" + str(new_balance) + ":"
        with open("data.txt", "w") as data
            data.writelines(data_content)
        OUTPUT(" You have successfully withdraw your balance!")
         f = open("report.txt", "a")
        WRITE(userData[1] + "|Withdrawal RM" + str(withdraw_amt) + "|" + date() + "|The current balance is now "
            + str(new_balance) + "") INTO F
        CLOSE F
        customer_menu(userData, lineNumber)
    ENDWHILE
ENDDEFINE
```

TO PERFORM TRANSACTION (DEPOSIT)

```
DEFINE FUNCTION customer_deposit(userData, lineNumber)
DECLARE balance,deposit_amt,new_balance as integer
     balance = int(userData[6])
           DECLARE deposit_amt as integer
           DISPLAY(" How much money you would like to deposit "))
           READ deposit_amt
     new_balance = 0
    DOWHILE TRUE
         new_balance = balance + deposit_amt
        with open("data.txt", "r") as data
             data_content = data.readlines()
         data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + userData[2] + ":" + userData[3] + ":" +
              userData[4] + ":" + userData[5] + ":" + str(new_balance) + ":"
        with open("data.txt", "w") as data
            data.writelines(data_content)
        OUTPUT(" You have successfully deposit your balance!")
         f = open("report.txt", "a")
        WRITE(userData[1] + "|Deposit RM" + str(deposit_amt) + "|" + date() + "|The current balance is now "
                + str(new_balance) + "") in= f
        CLOSE F
        customer_menu(userData, lineNumber)
ENDDEFINE
```

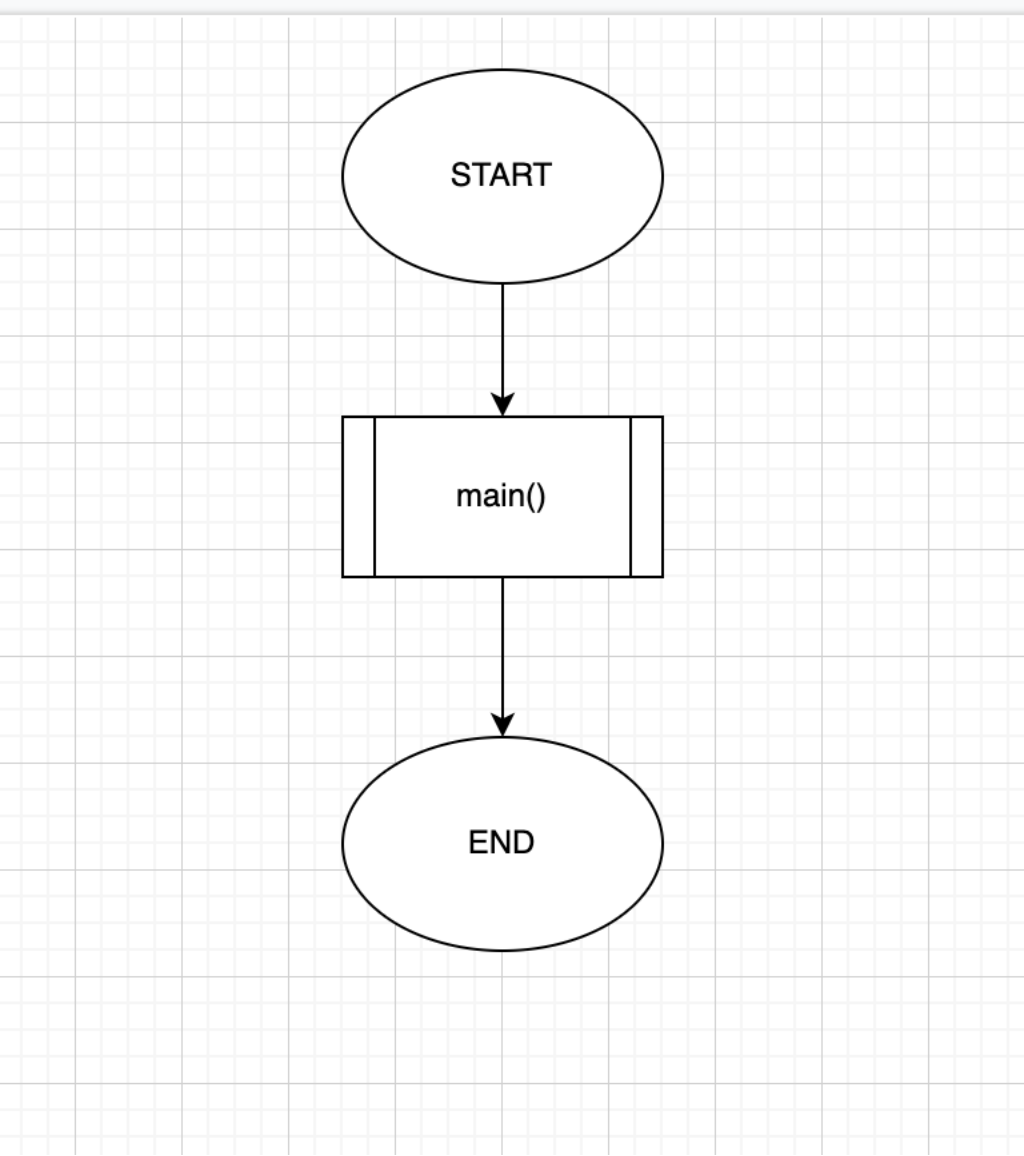
TO PERFORM TRANSACTION(TRANSFER)

```
    DOWHILE TRUE
        IF balance >= transfer_amt THEN
             new_balance = balance - transfer_amt
             new_dest_balance = dest_balance + transfer_amt
            break
        ELSE
            OUTPUT(" Your balance is not enough = transfer  ")
            DECLARE withdraw_amt as transfer
            DISPLAY(" How much money you would like to transfer "))
            READ transfer_amt
        ENDIF    with open("data.txt", "r") as data
         data_content = data.readlines()
         data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + userData[2] + ":" + userData[3] + ":" +
           userData[4] + ":" + userData[5] + ":" + str(new_balance) + ":"
    with open("data.txt", "w") as data
        data.writelines(data_content)
    OUTPUT(" You have successfully transfer your balance!")    with open("data.txt", "r") as data
         data_content = data.readlines()
     data_content[lineDest] = DestData[0] + ":" + DestData[1] + ":" + DestData[2] + ":" + DestData[3] + ":" +
        DestData[4] + ":" + DestData[5] + ":" + str(new_dest_balance) + ":"
    with open("data.txt", "w") as data
        data.writelines(data_content)
     f = open("report.txt", "a")
    WRITE(userData[1] + "|Transfer RM" + str(
        transfer_amt) + "|" + date() + "|The current balance is now " + str(new_balance) + "")
    WRITE(accDest + "|have successfully receive RM" + str(transfer_amt) + "|" + date() + "|The current balance is now "
        + str(new_dest_balance) + "") into f
    CLOSE F
    customer_menu(userData, lineNumber)
ENDDEFINE
```

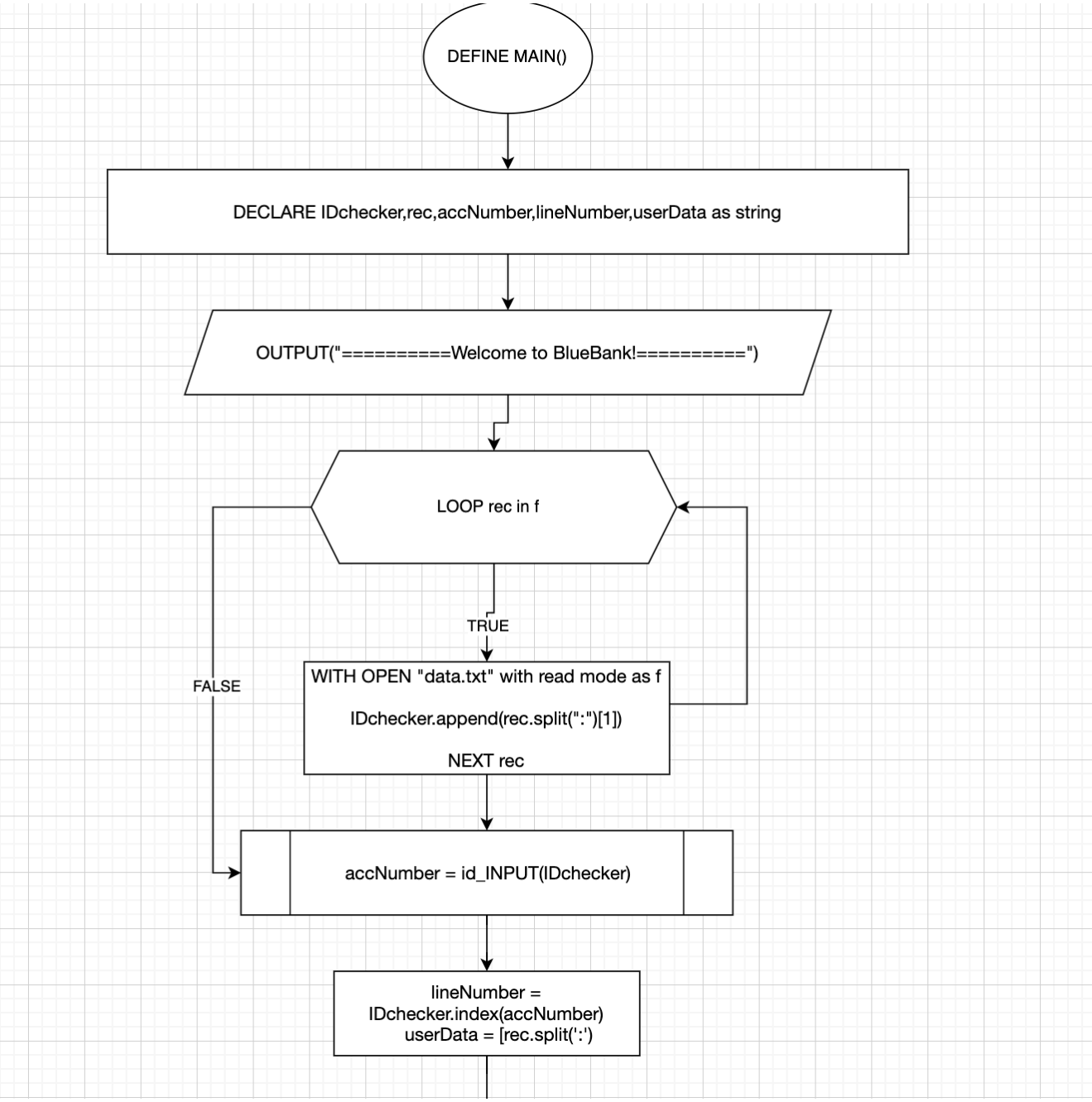
```
DEFINE FUNCTION customer_transfer(userData, lineNumber)
DECLARE balance,new_balance,new_dest_balance as integerr
     balance = int(userData[6])
     dest_IDchecker = []
    LOOP rec IN open("data.txt").readlines()
        dest_IDchecker.append(rec.split(":")[1])
    NEXT rec
     accDest = dest_displaydest_IDchecker)
    OUTPUT(dest_IDchecker)
    OUTPUT(accDest)
     lineDest = dest_IDchecker.index(accDest)
     DestData = [rec.split(':') LOOP rec IN open("data.txt").readlines()][lineDest]
    NEXT rec
            DECLARE withdraw_amt as transfer
            DISPLAY(" How much money you would like to transfer "))
            READ transfer_amt
     new_balance = 0
     dest_balance = int(DestData[6])
     new_dest_balance = 0
```

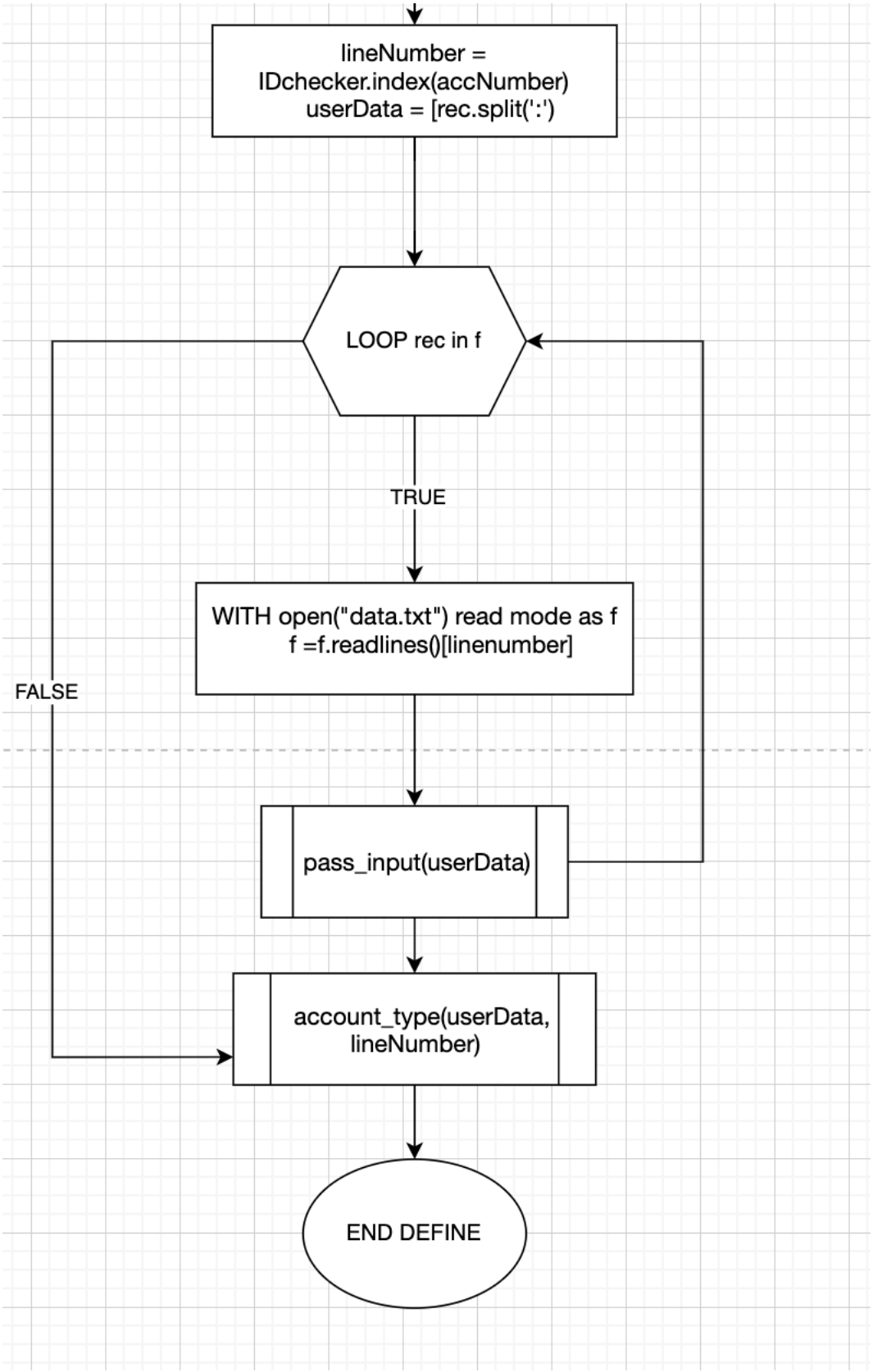
Design of program – flowchart

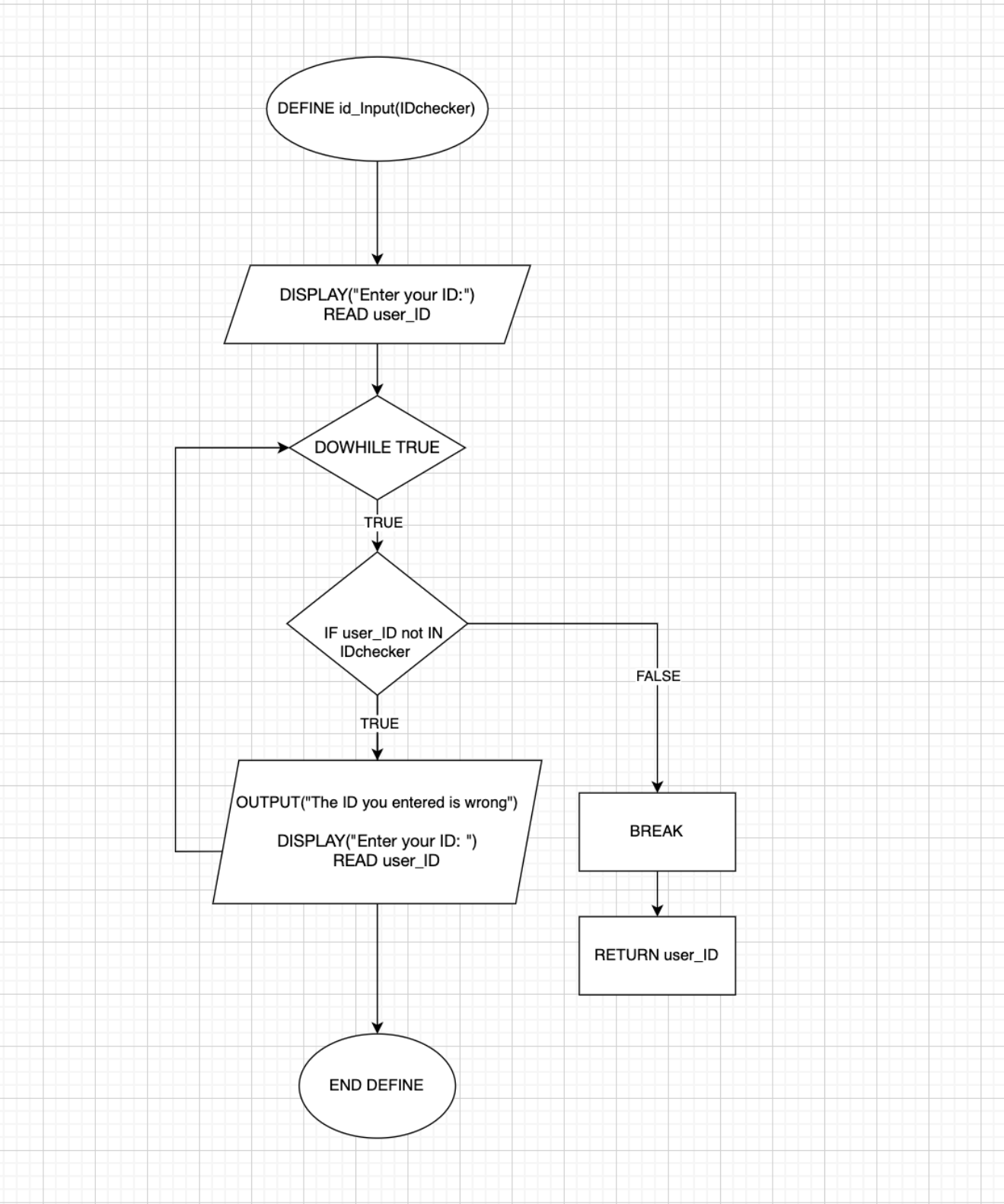
MAIN LOGIC

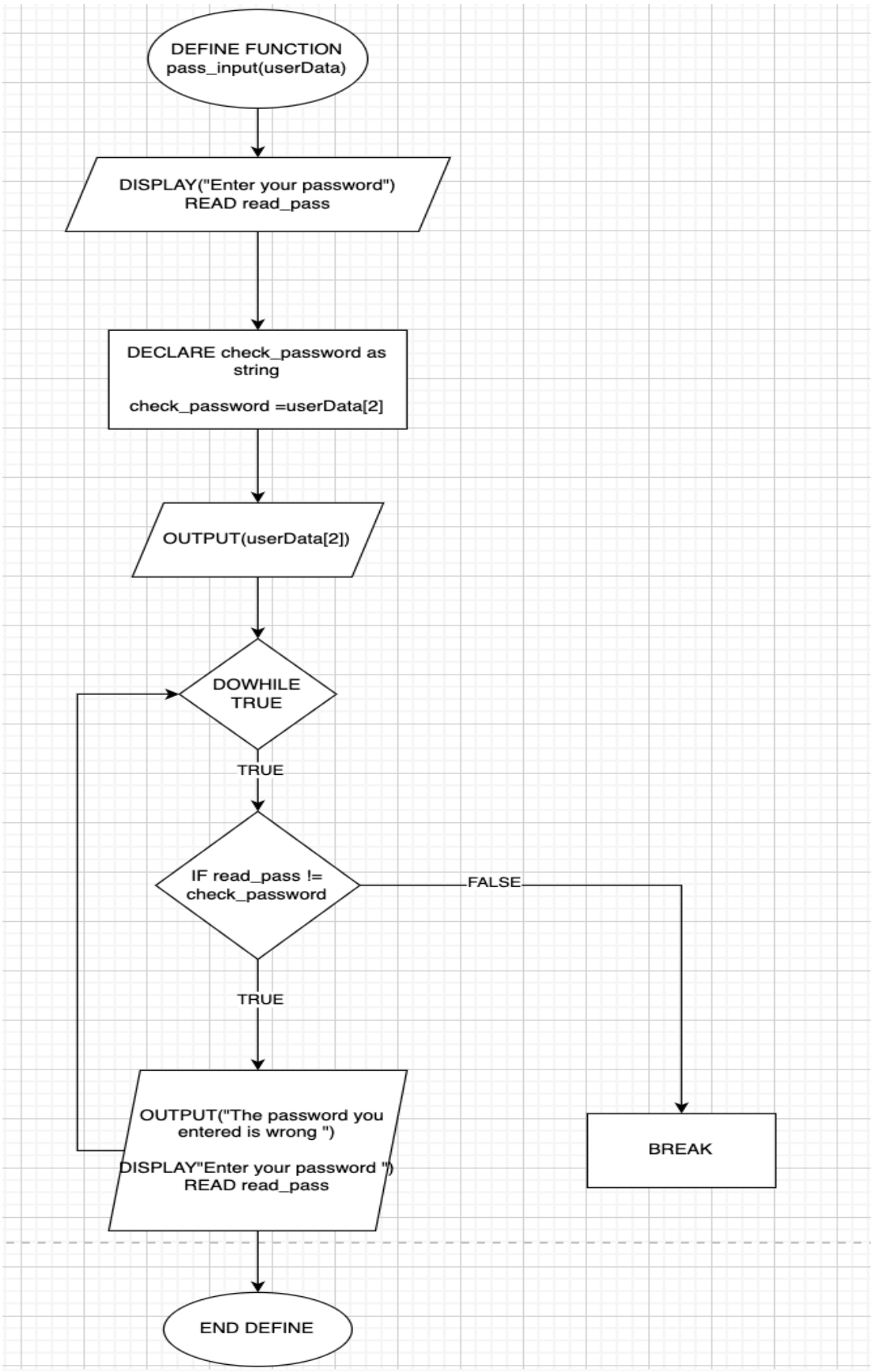


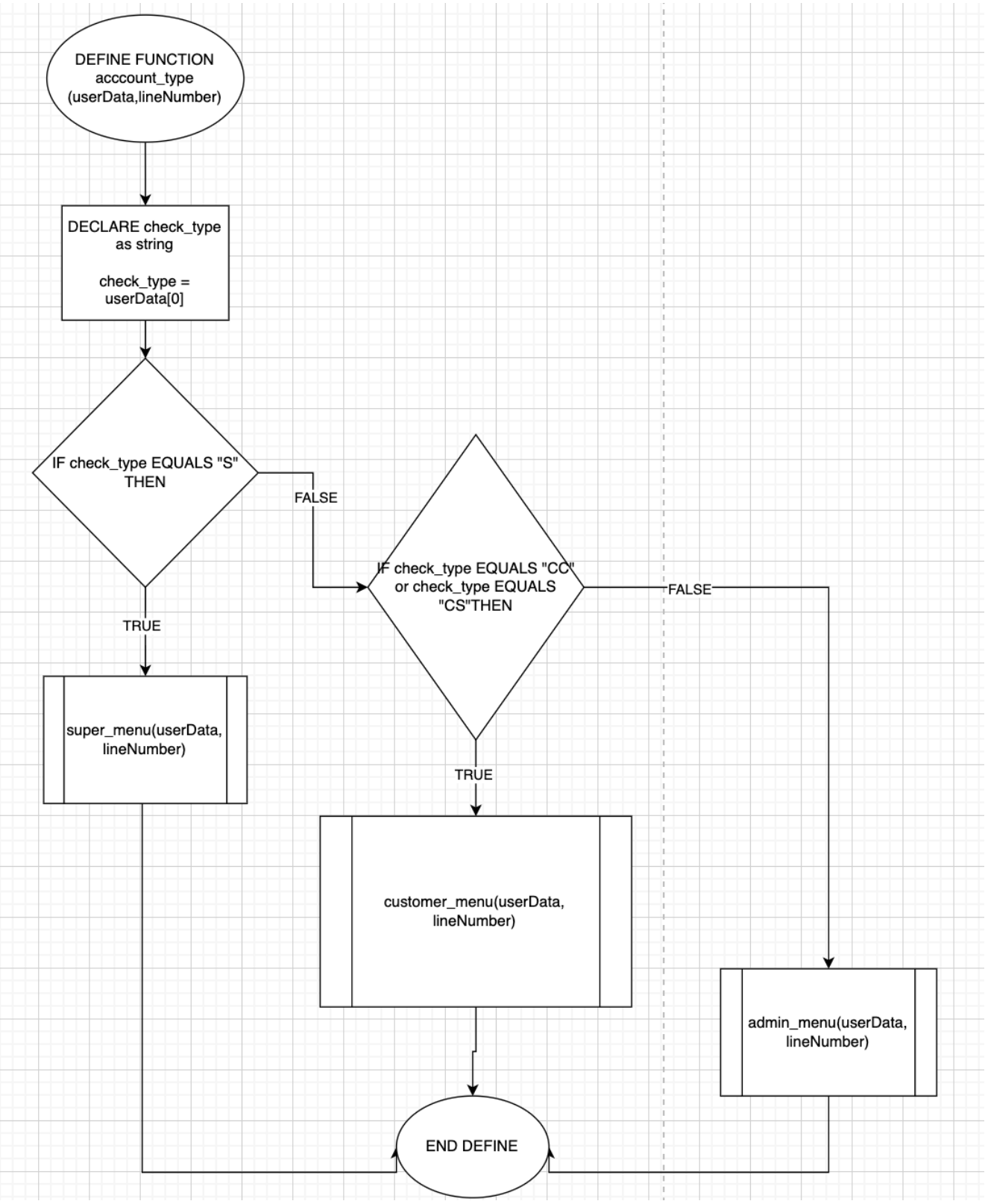
LOGIN MENU


DEFINE MAIN()
DECLARE IDchecker,rec,accNumber,lineNumber,userData as string
OUTPUT("==========Welcome to BlueBank!==========")
LOOP rec in f
TRUE
FALSE
WITH OPEN "data.txt" with read mode as f
IDchecker.append(rec.split(":")[1])
NEXT rec
accNumber = id_INPUT(IDchecker)
lineNumber =
IDchecker.index(accNumber)
userData = [rec.split(':')

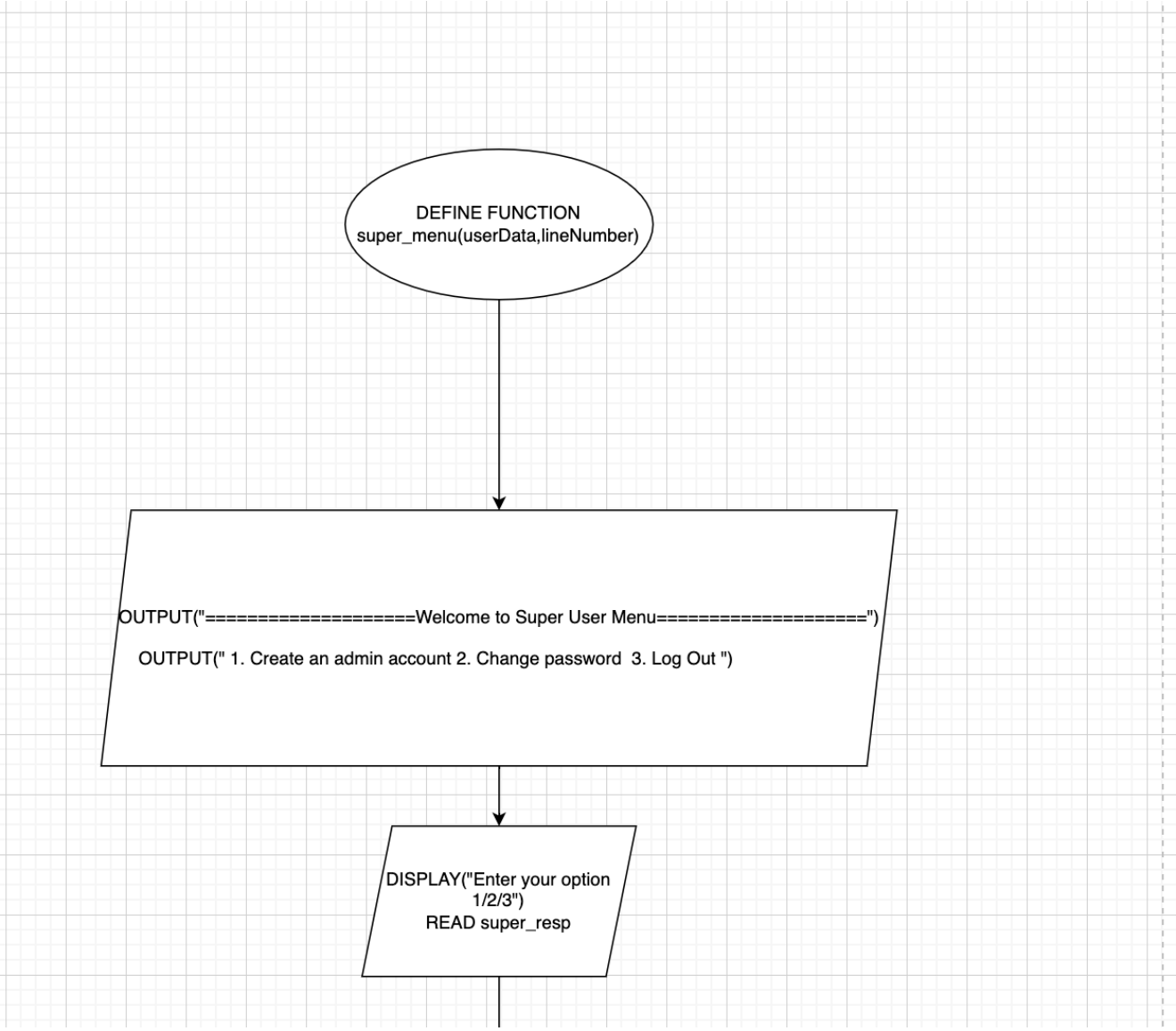

lineNumber =
IDchecker.index(accNumber)
userData = [rec.split(':')
LOOP rec in f
TRUE
WITH open("data.txt") read mode as f
f =f.readlines()[linenumber]
FALSE
pass_input(userData)
account_type(userData,
lineNumber)
END DEFINE

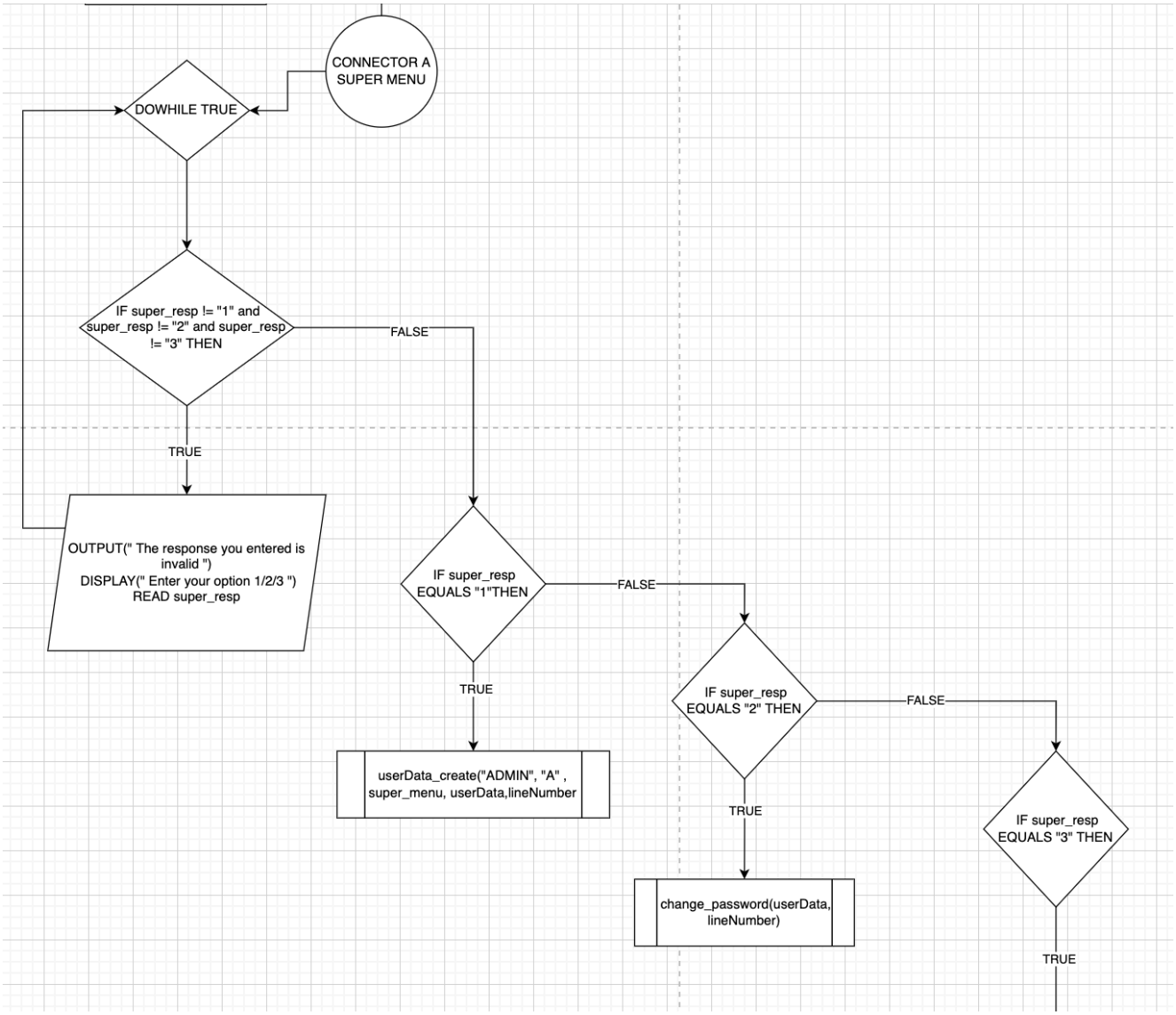

DEFINE id_Input(IDchecker)
DISPLAY("Enter your ID:")
READ user_ID
DOWHILE TRUE
TRUE
IF user_ID not IN IDchecker
FALSE
TRUE
OUTPUT("The ID you entered is wrong")
DISPLAY("Enter your ID: ")
READ user_ID
BREAK
RETURN user_ID
END DEFINE

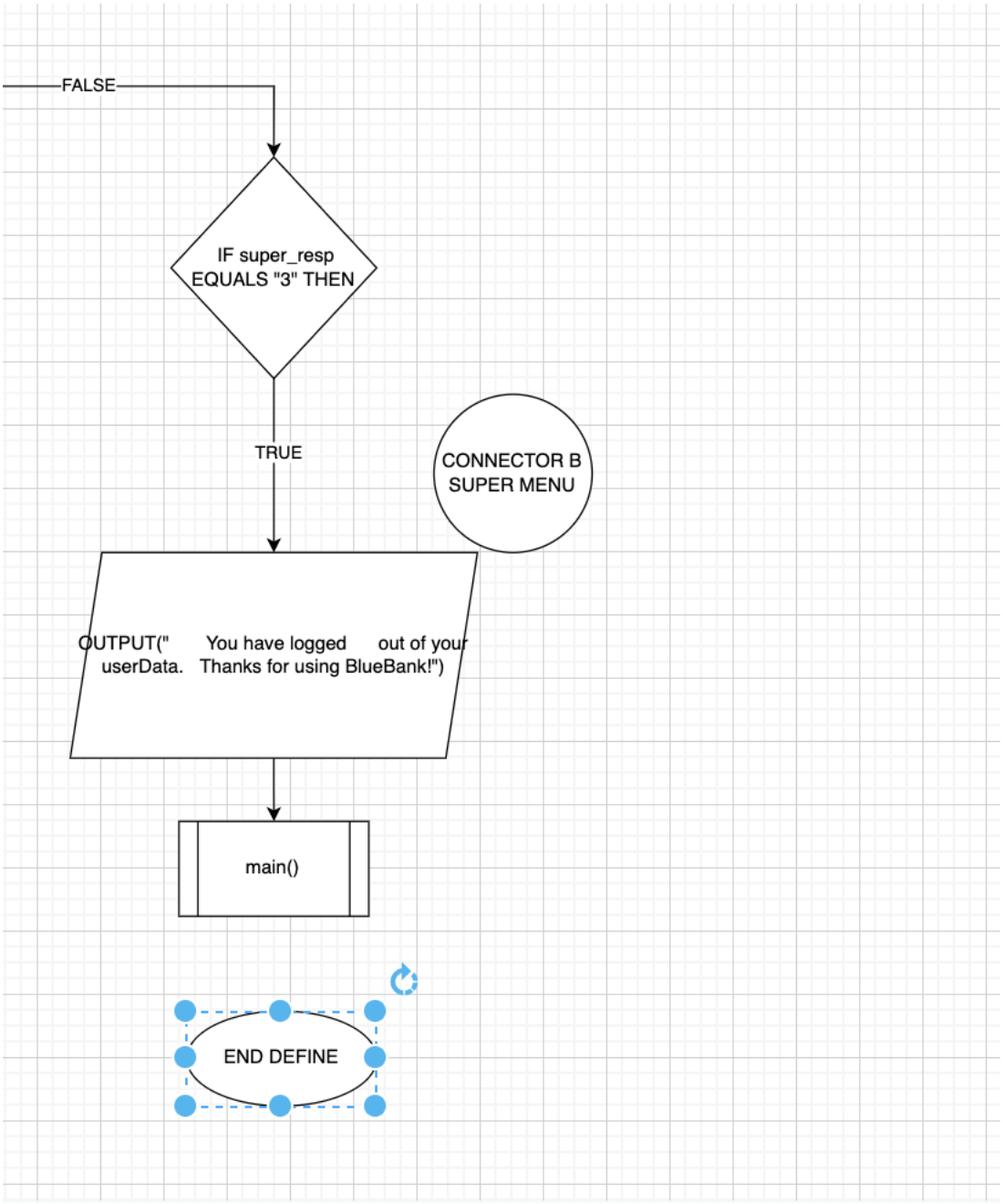

DEFINE FUNCTION
pass_input(userData)
DISPLAY("Enter your password")
READ read_pass
DECLARE check_password as string
check_password =userData[2]
OUTPUT(userData[2])
DOWHILE TRUE
TRUE
IF read_pass != check_password
FALSE
TRUE
OUTPUT("The password you entered is wrong ")
DISPLAY"Enter your password ")
READ read_pass
BREAK
END DEFINE


DEFINE FUNCTION acccount_type (userData,lineNumber)
DECLARE check_type as string
check_type = userData[0]
IF check_type EQUALS "S" THEN
FALSE
IF check_type EQUALS "CC" or check_type EQUALS "CS"THEN
FALSE
TRUE
super_menu(userData, lineNumber)
TRUE
customer_menu(userData, lineNumber)
admin_menu(userData, lineNumber)
END DEFINE

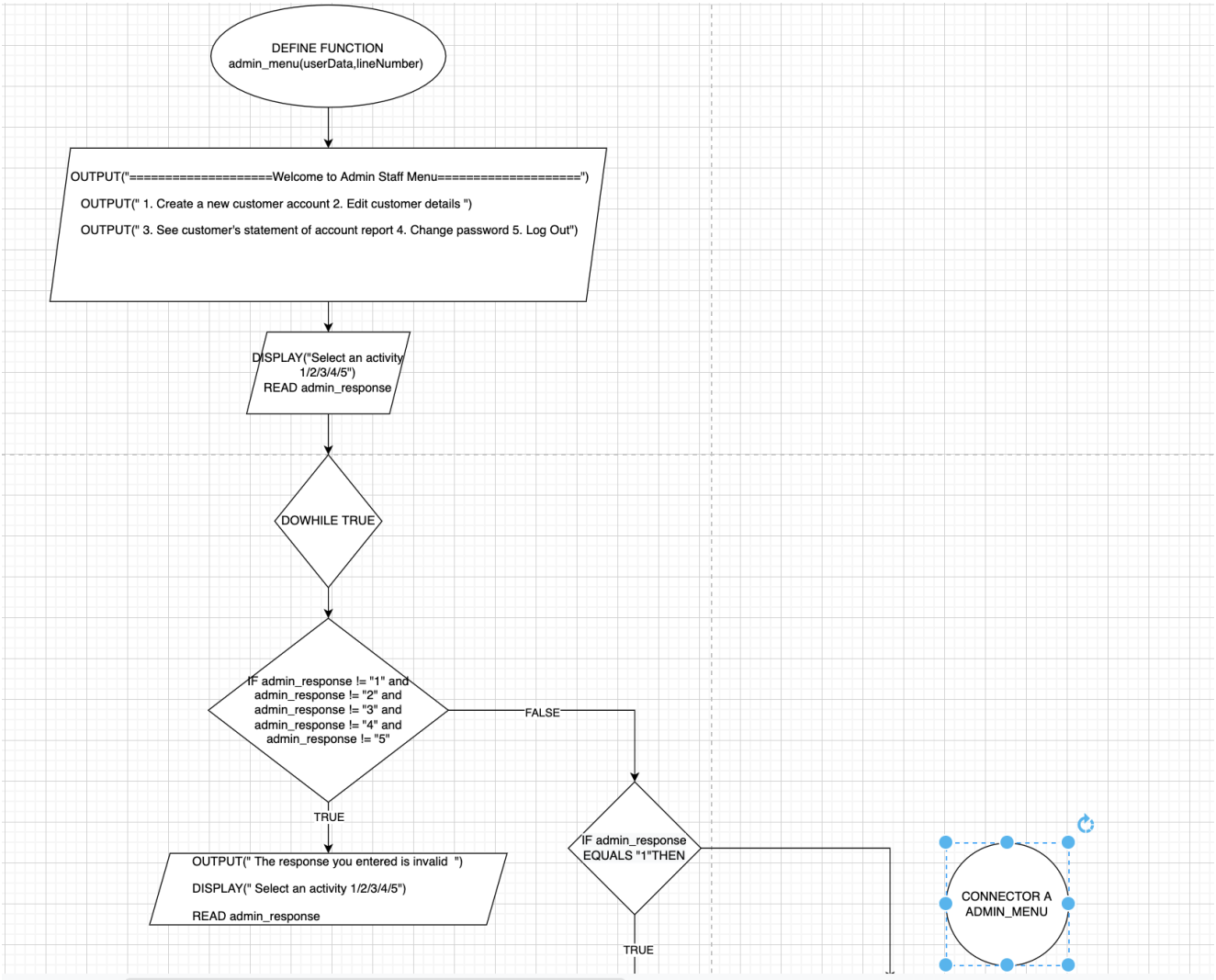
SUPER MENU

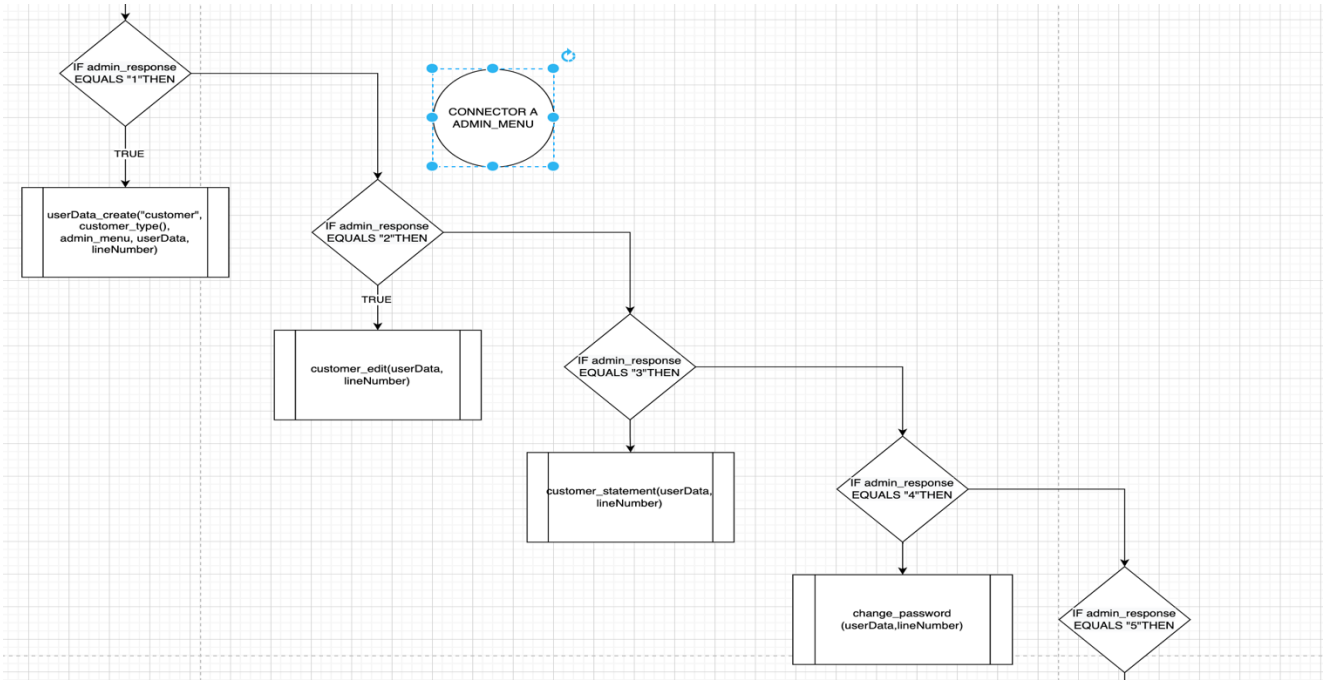

DEFINE FUNCTION
super_menu(userData,lineNumber)
OUTPUT("====================Welcome to Super User Menu====================")
OUTPUT(" 1. Create an admin account 2. Change password 3. Log Out ")
DISPLAY("Enter your option 1/2/3")
READ super_resp

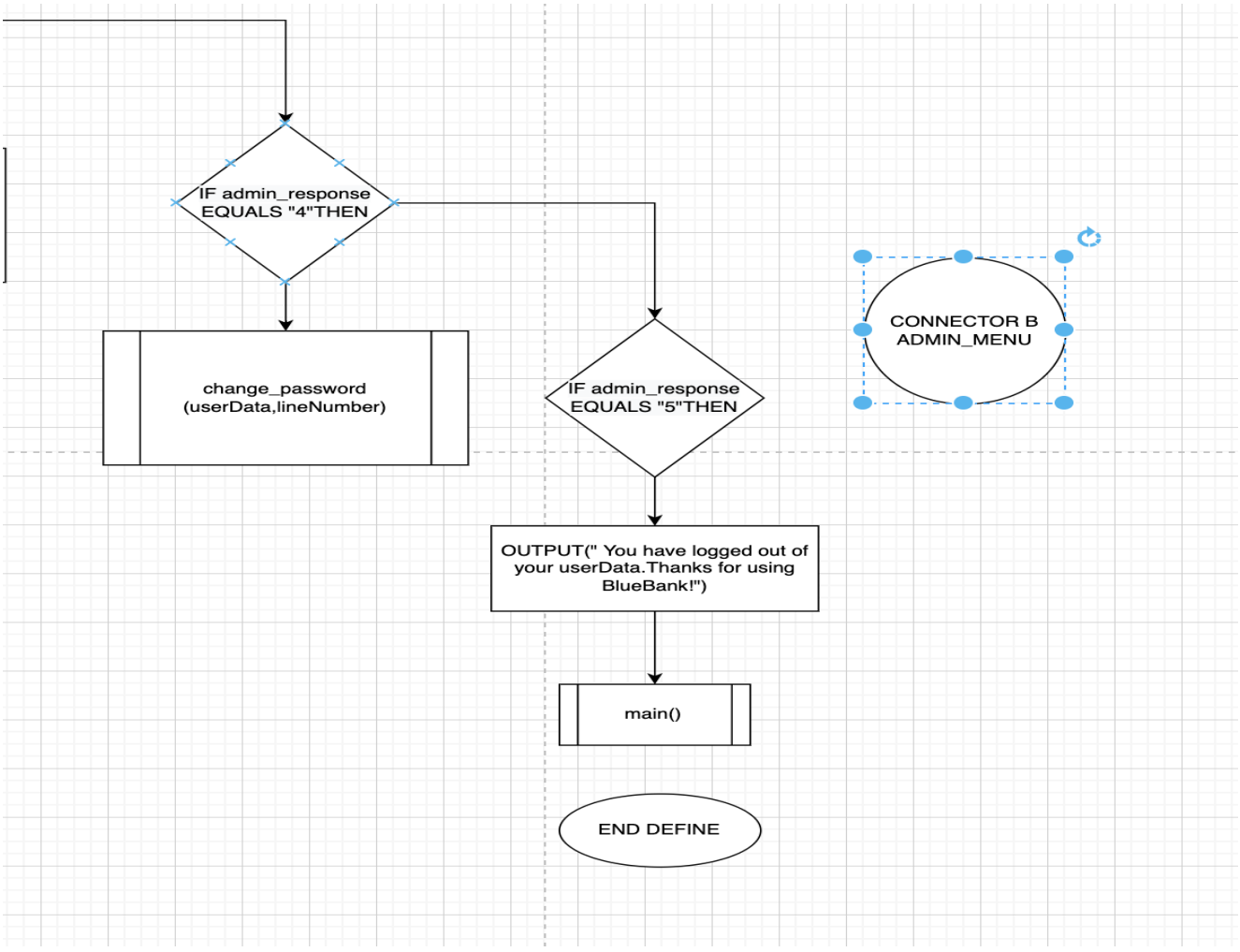

CONNECTOR A
SUPER MENU
DOWHILE TRUE
IF super_resp != "1" and super_resp != "2" and super_resp != "3" THEN
FALSE
TRUE
OUTPUT(" The response you entered is invalid ")
DISPLAY(" Enter your option 1/2/3 ")
READ super_resp
IF super_resp EQUALS "1"THEN
FALSE
TRUE
userData_create("ADMIN", "A" , super_menu, userData,lineNumber
IF super_resp EQUALS "2" THEN
FALSE
TRUE
change_password(userData, lineNumber)
IF super_resp EQUALS "3" THEN
TRUE


FALSE
IF super_resp
EQUALS "3" THEN
TRUE
CONNECTOR B
SUPER MENU
OUTPUT(" You have logged out of your
userData. Thanks for using BlueBank!")
main()
END DEFINE

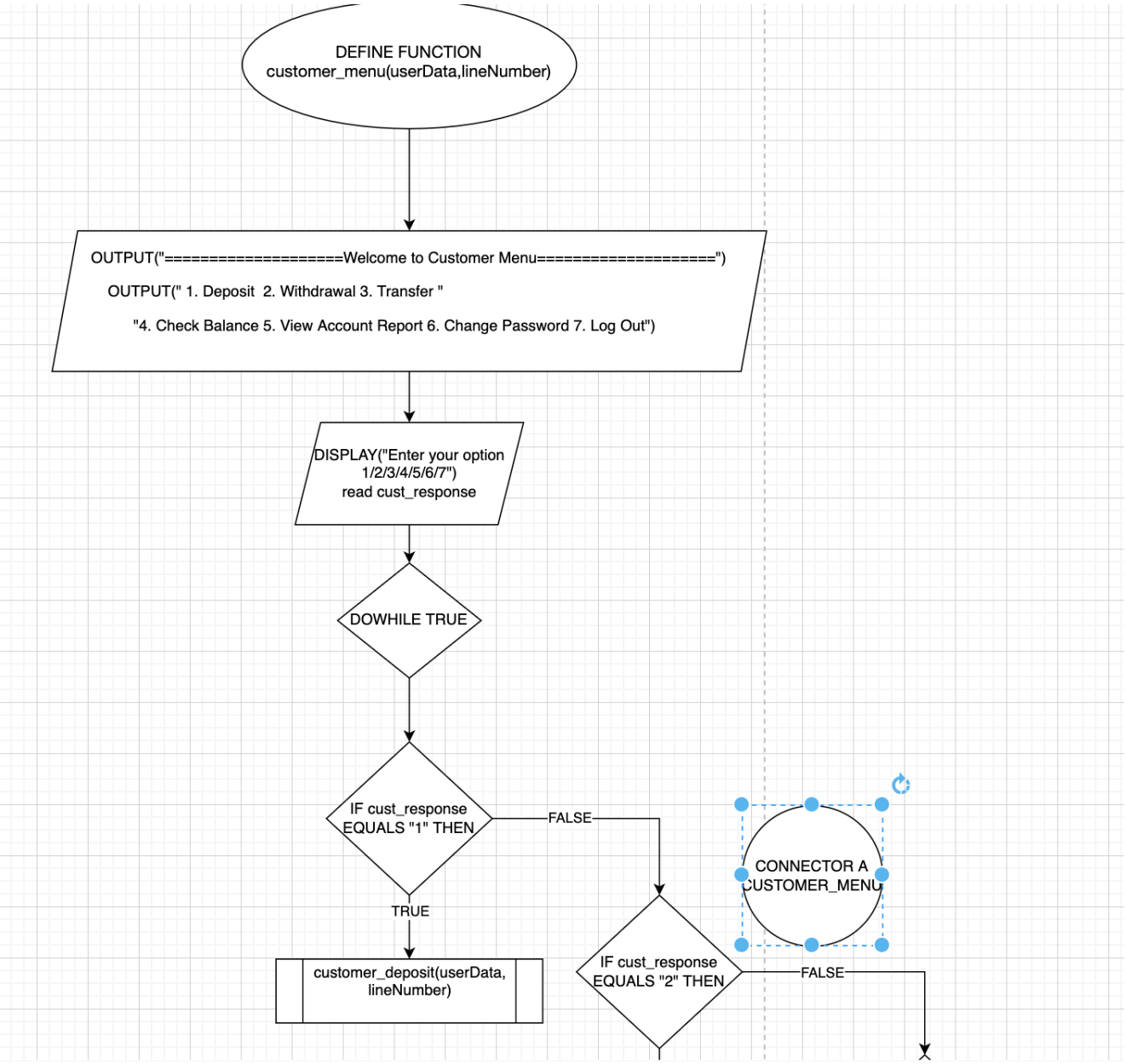
ADMIN MENU

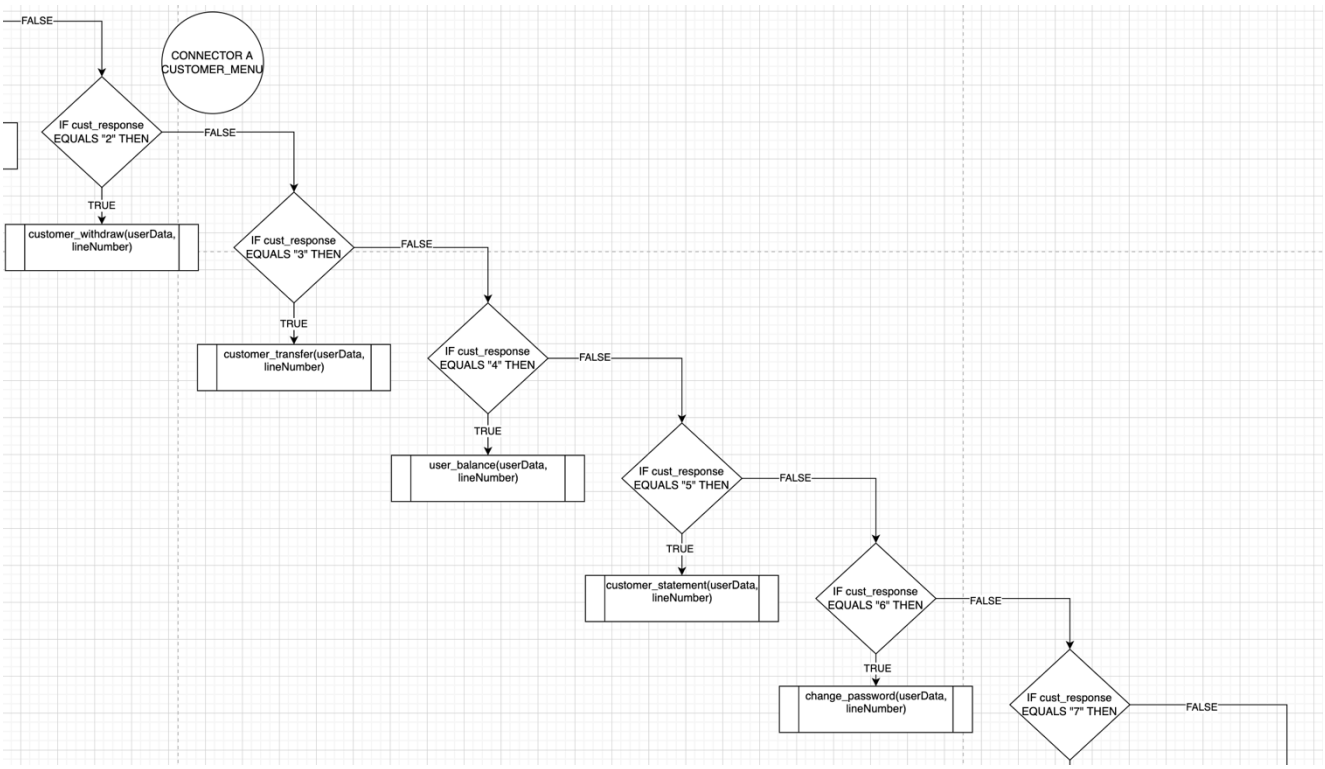

DEFINE FUNCTION
admin_menu(userData,lineNumber)
OUTPUT("====================Welcome to Admin Staff Menu====================")
OUTPUT(" 1. Create a new customer account 2. Edit customer details ")
OUTPUT(" 3. See customer's statement of account report 4. Change password 5. Log Out")
DISPLAY("Select an activity 1/2/3/4/5")
READ admin_response
DOWHILE TRUE
IF admin_response != "1" and admin_response != "2" and admin_response != "3" and admin_response != "4" and admin_response != "5"
FALSE
TRUE
OUTPUT(" The response you entered is invalid ")
DISPLAY(" Select an activity 1/2/3/4/5")
READ admin_response
IF admin_response EQUALS "1"THEN
TRUE
CONNECTOR A
ADMIN_MENU

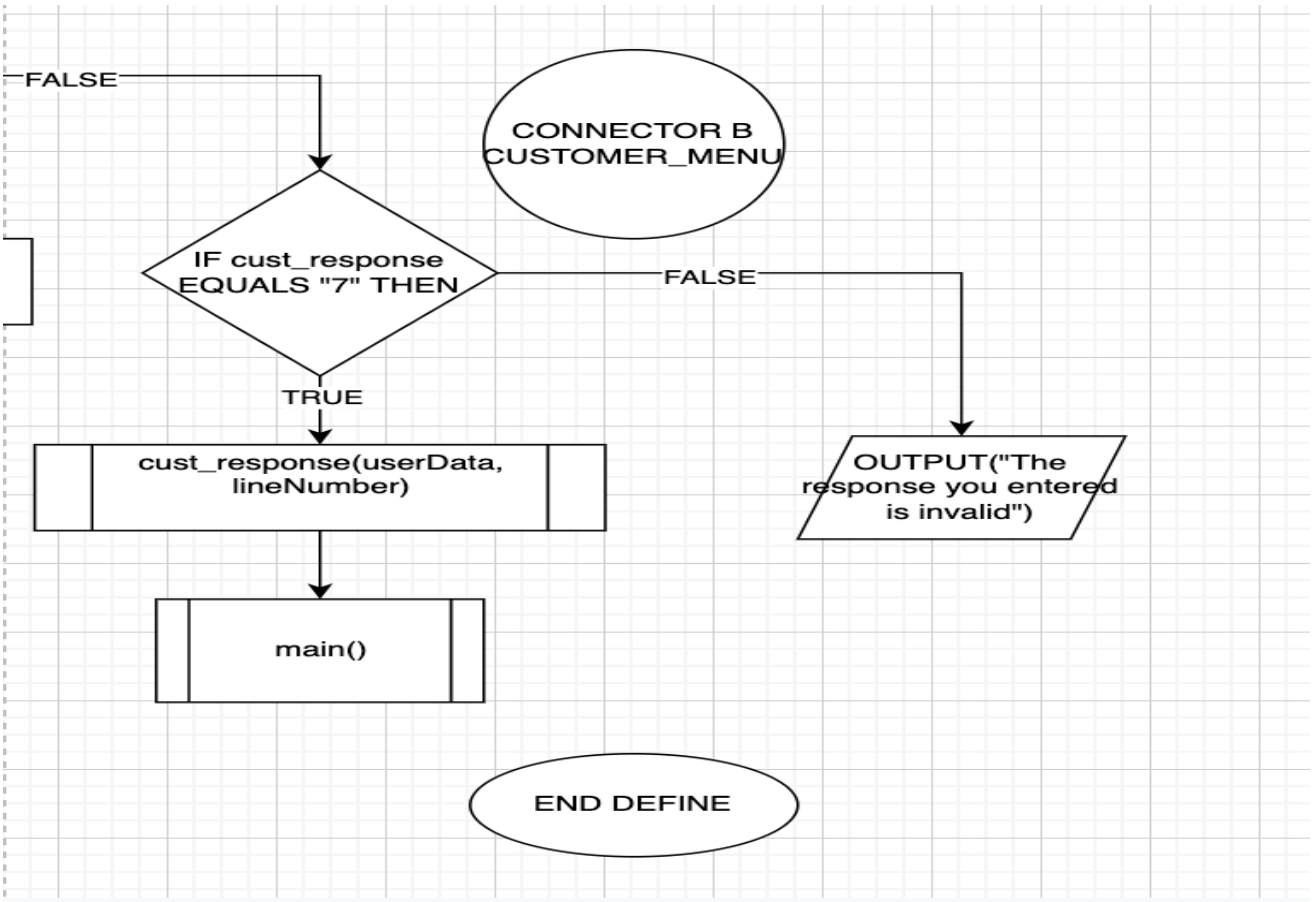

IF admin_response EQUALS "1"THEN
TRUE
userData_create("customer", customer_type(), admin_menu, userData, lineNumber)
CONNECTOR A ADMIN_MENU
IF admin_response EQUALS "2"THEN
TRUE
customer_edit(userData, lineNumber)
IF admin_response EQUALS "3"THEN
customer_statement(userData, lineNumber)
IF admin_response EQUALS "4"THEN
change_password (userData,lineNumber)
IF admin_response EQUALS "5"THEN


IF admin_response EQUALS "4"THEN
change_password (userData,lineNumber)
IF admin_response EQUALS "5"THEN
CONNECTOR B ADMIN_MENU
OUTPUT(" You have logged out of your userData.Thanks for using BlueBank!")
main()
END DEFINE

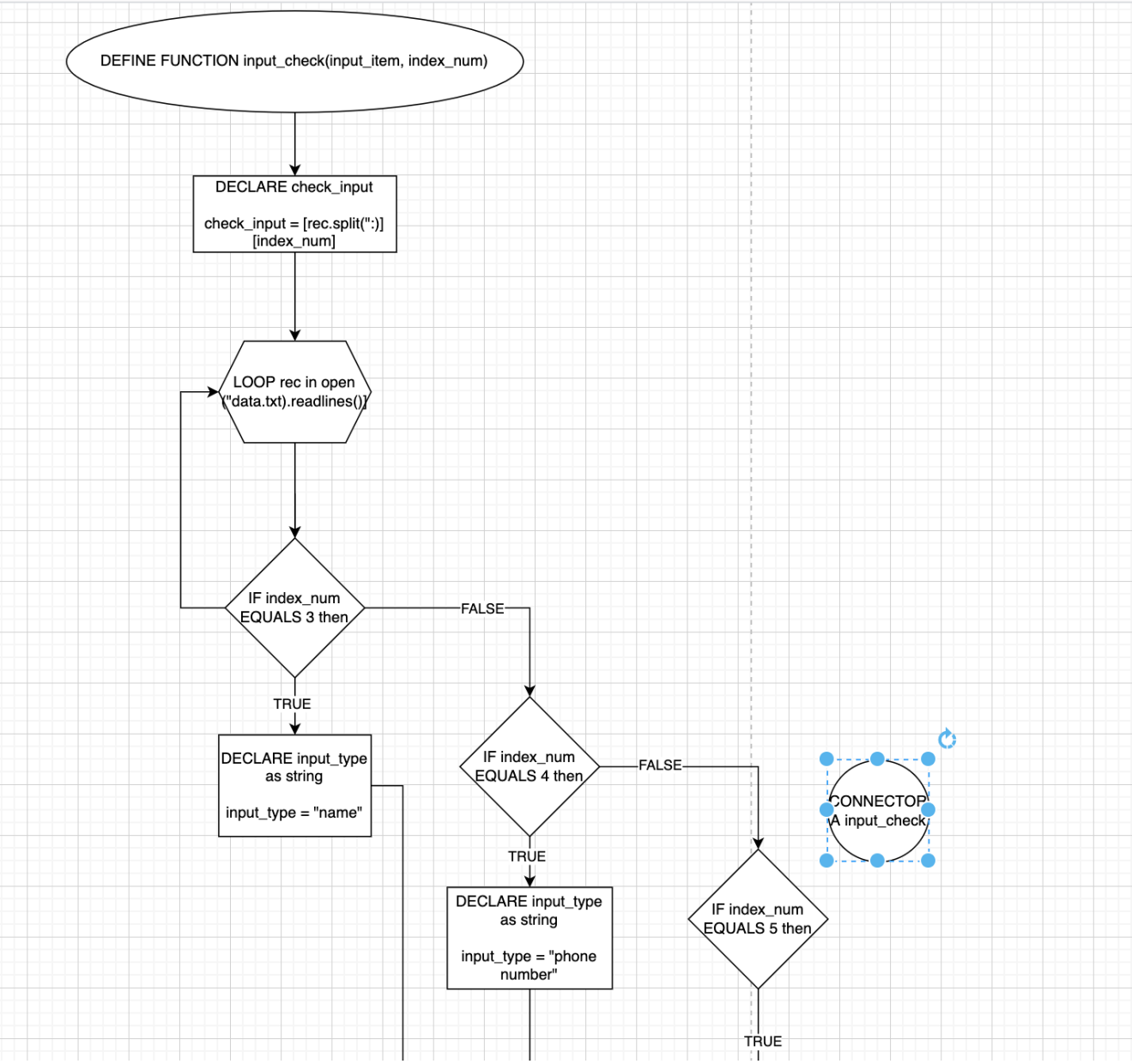
CUSTOMER MENU

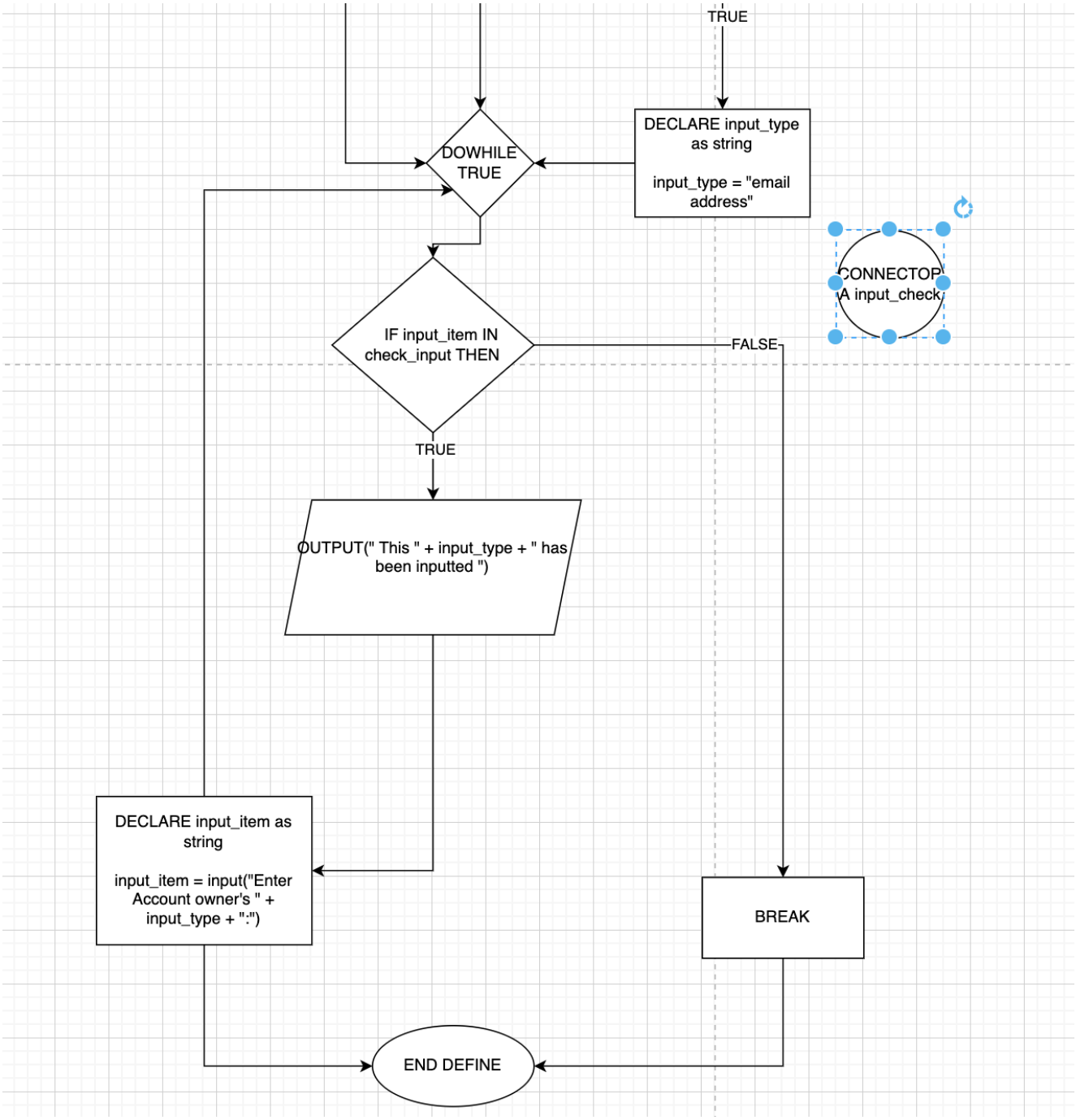



FALSE
CONNECTOR A
CUSTOMER_MENU
IF cust_response EQUALS "2" THEN
FALSE
TRUE
customer_withdraw(userData, lineNumber)
IF cust_response EQUALS "3" THEN
FALSE
TRUE
customer_transfer(userData, lineNumber)
IF cust_response EQUALS "4" THEN
FALSE
TRUE
user_balance(userData, lineNumber)
IF cust_response EQUALS "5" THEN
FALSE
TRUE
customer_statement(userData, lineNumber)
IF cust_response EQUALS "6" THEN
FALSE
TRUE
change_password(userData, lineNumber)
IF cust_response EQUALS "7" THEN
FALSE

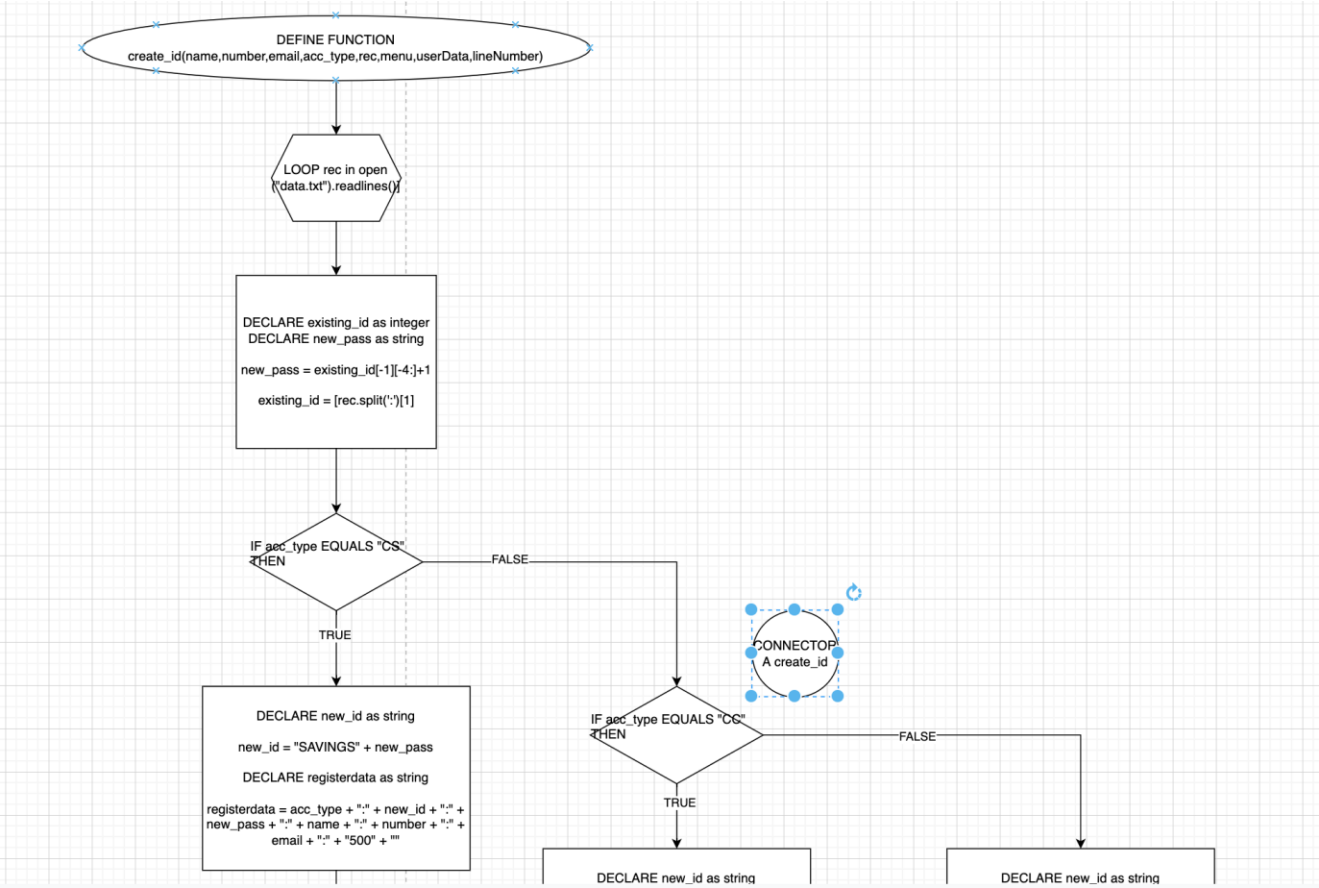

FALSE
CONNECTOR B
CUSTOMER_MENU
IF cust_response EQUALS "7" THEN
FALSE
TRUE
cust_response(userData, lineNumber)
OUTPUT("The response you entered is invalid")
main()
END DEFINE

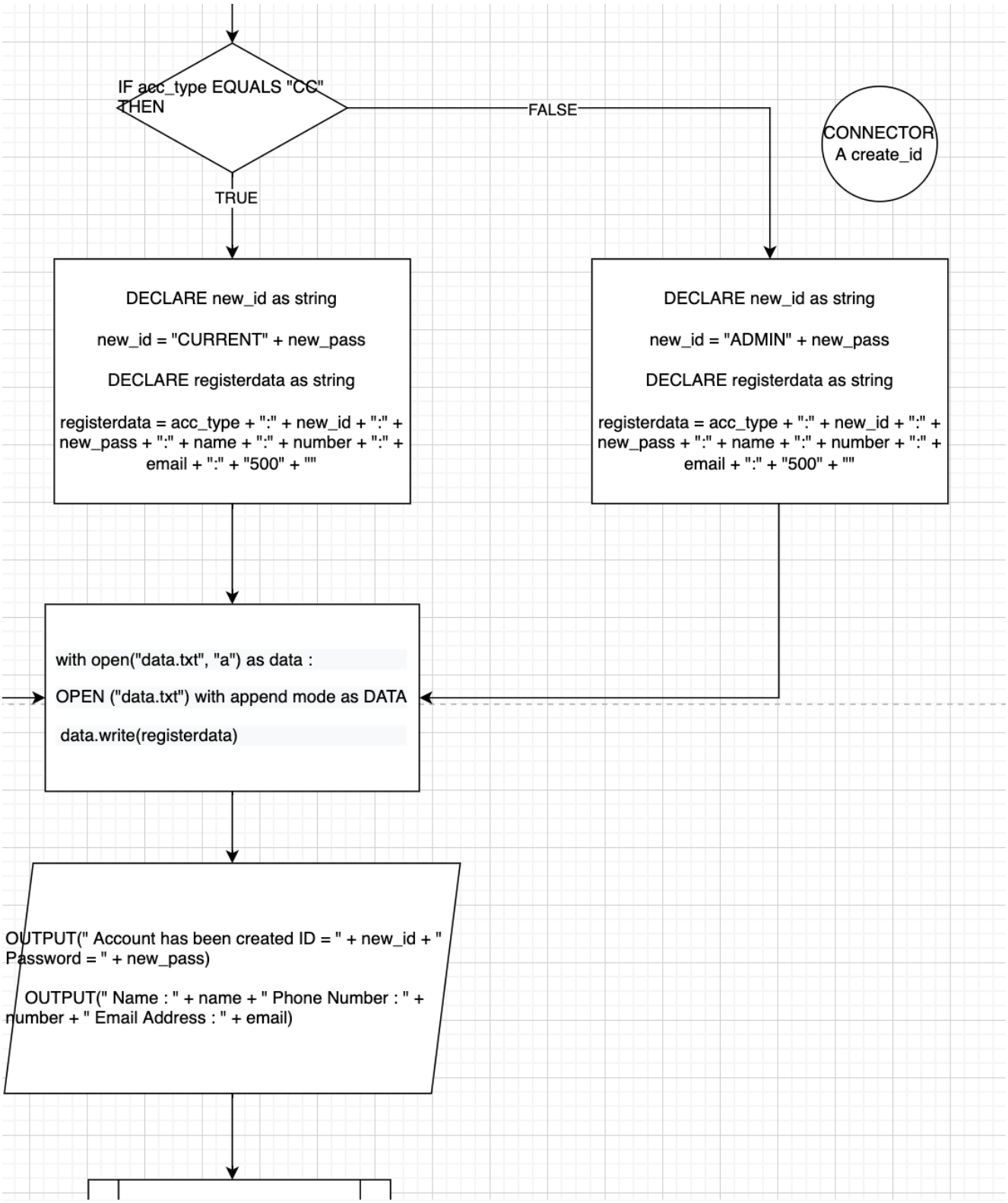
INPUT_CHECK

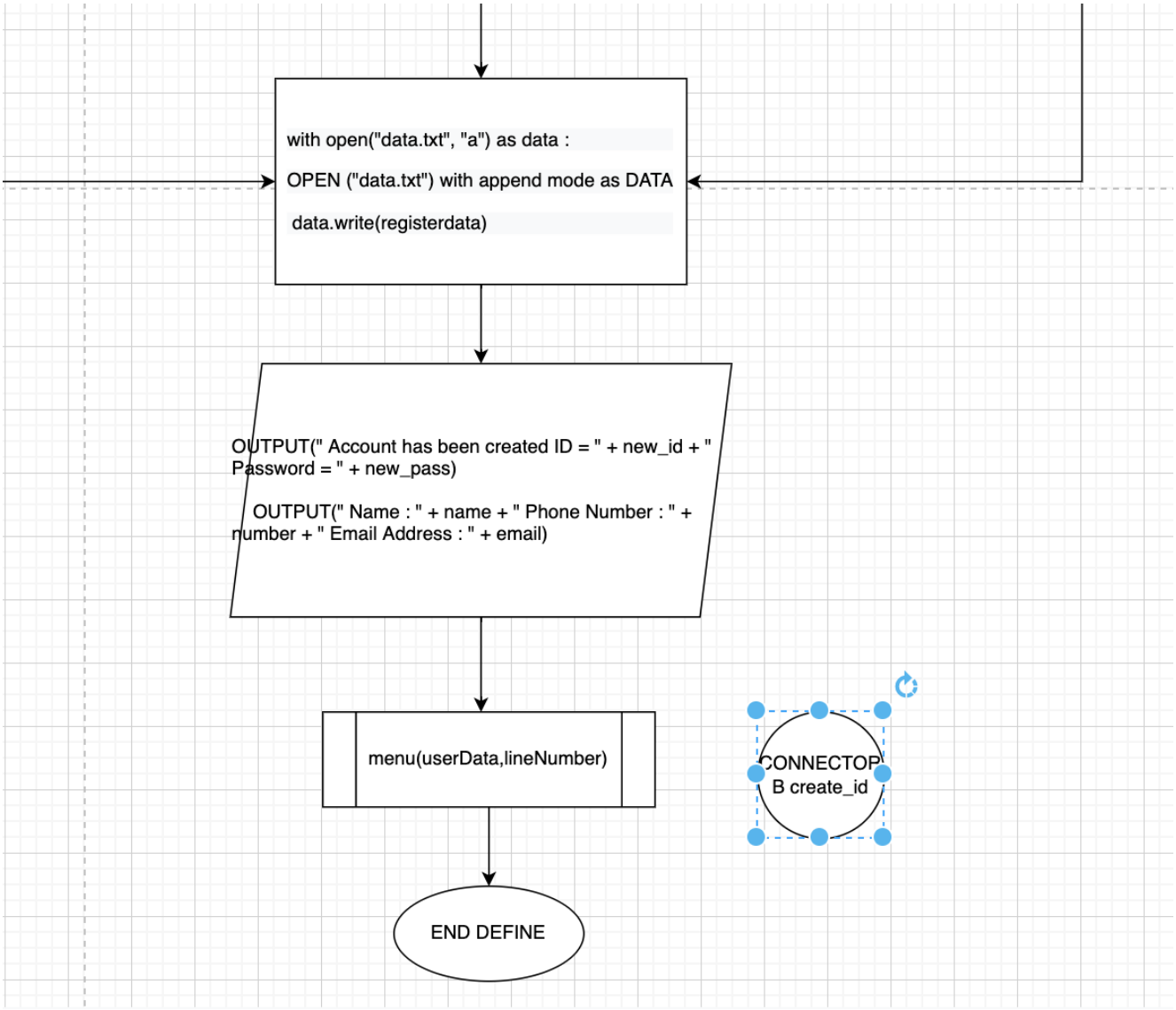

DEFINE FUNCTION input_check(input_item, index_num)
DECLARE check_input
check_input = [rec.split(":)] [index_num]
LOOP rec in open ("data.txt).readlines())
IF index_num EQUALS 3 then
FALSE
TRUE
DECLARE input_type as string
input_type = "name"
IF index_num EQUALS 4 then
FALSE
TRUE
DECLARE input_type as string
input_type = "phone number"
IF index_num EQUALS 5 then
TRUE
CONNECTOR A input_check


TRUE
DOWHILE TRUE
DECLARE input_type as string
input_type = "email address"
CONNECTOR A input_check
IF input_item IN check_input THEN
FALSE
TRUE
OUTPUT(" This " + input_type + " has been inputted ")
DECLARE input_item as string
input_item = input("Enter Account owner's " + input_type + ":")
BREAK
END DEFINE

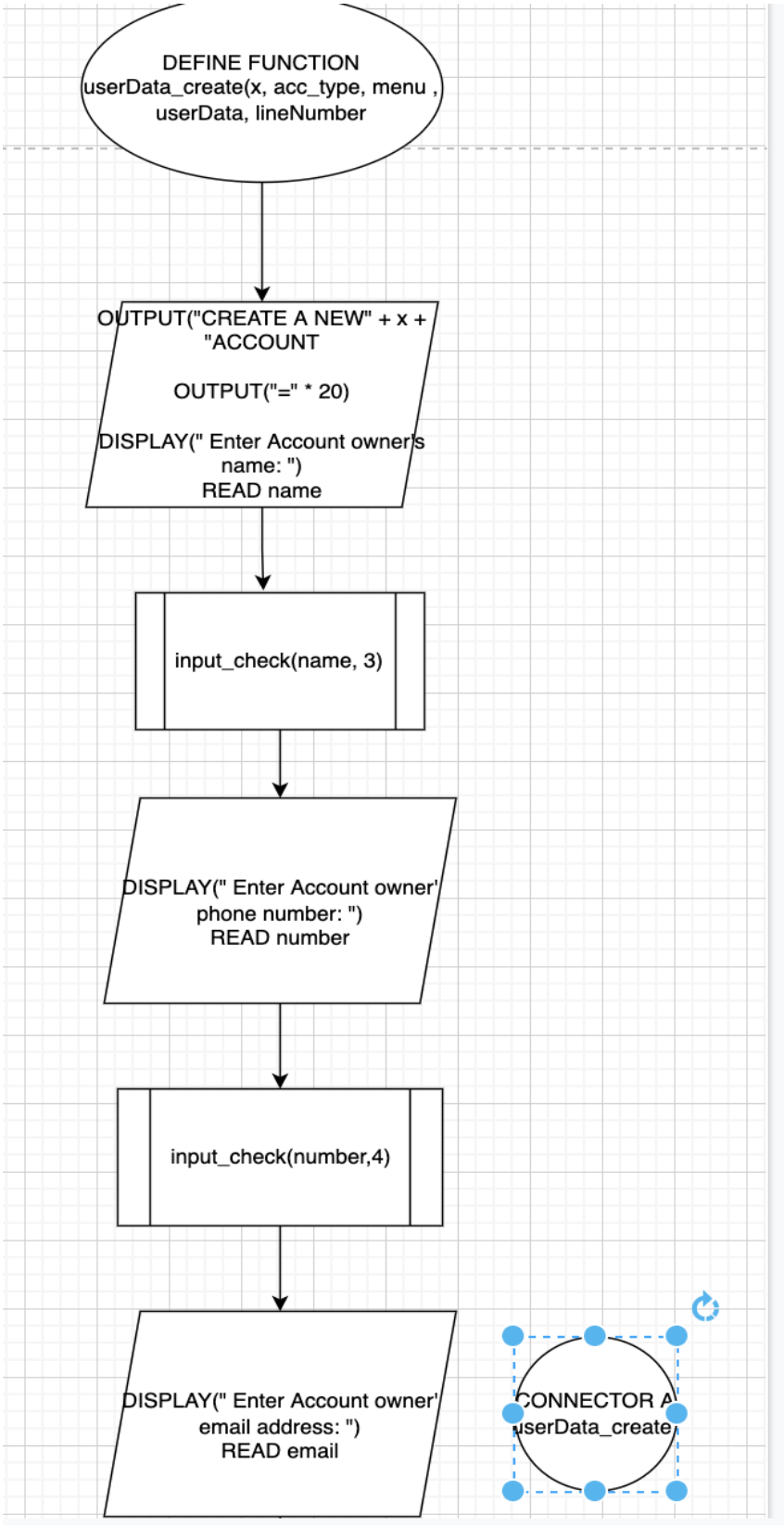
CREATE NEW ID

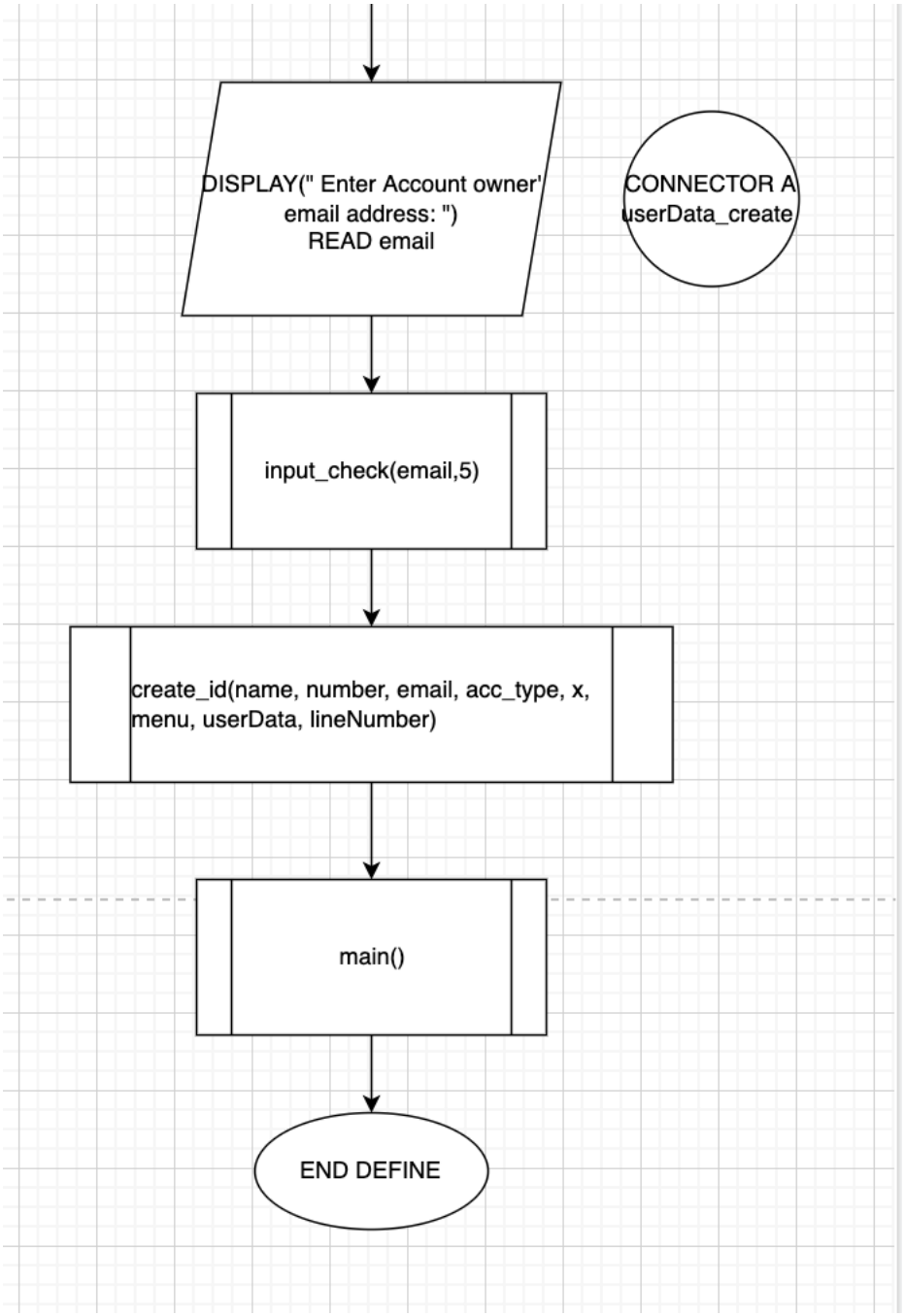



IF acc_type EQUALS "CC"
THEN
FALSE
TRUE
CONNECTOR
A create_id
DECLARE new_id as string
new_id = "CURRENT" + new_pass
DECLARE registerdata as string
registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ":" + "500" + ""
DECLARE new_id as string
new_id = "ADMIN" + new_pass
DECLARE registerdata as string
registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ":" + "500" + ""
with open("data.txt", "a") as data :
OPEN ("data.txt") with append mode as DATA
data.write(registerdata)
OUTPUT(" Account has been created ID = " + new_id + " Password = " + new_pass)
OUTPUT(" Name : " + name + " Phone Number : " + number + " Email Address : " + email)

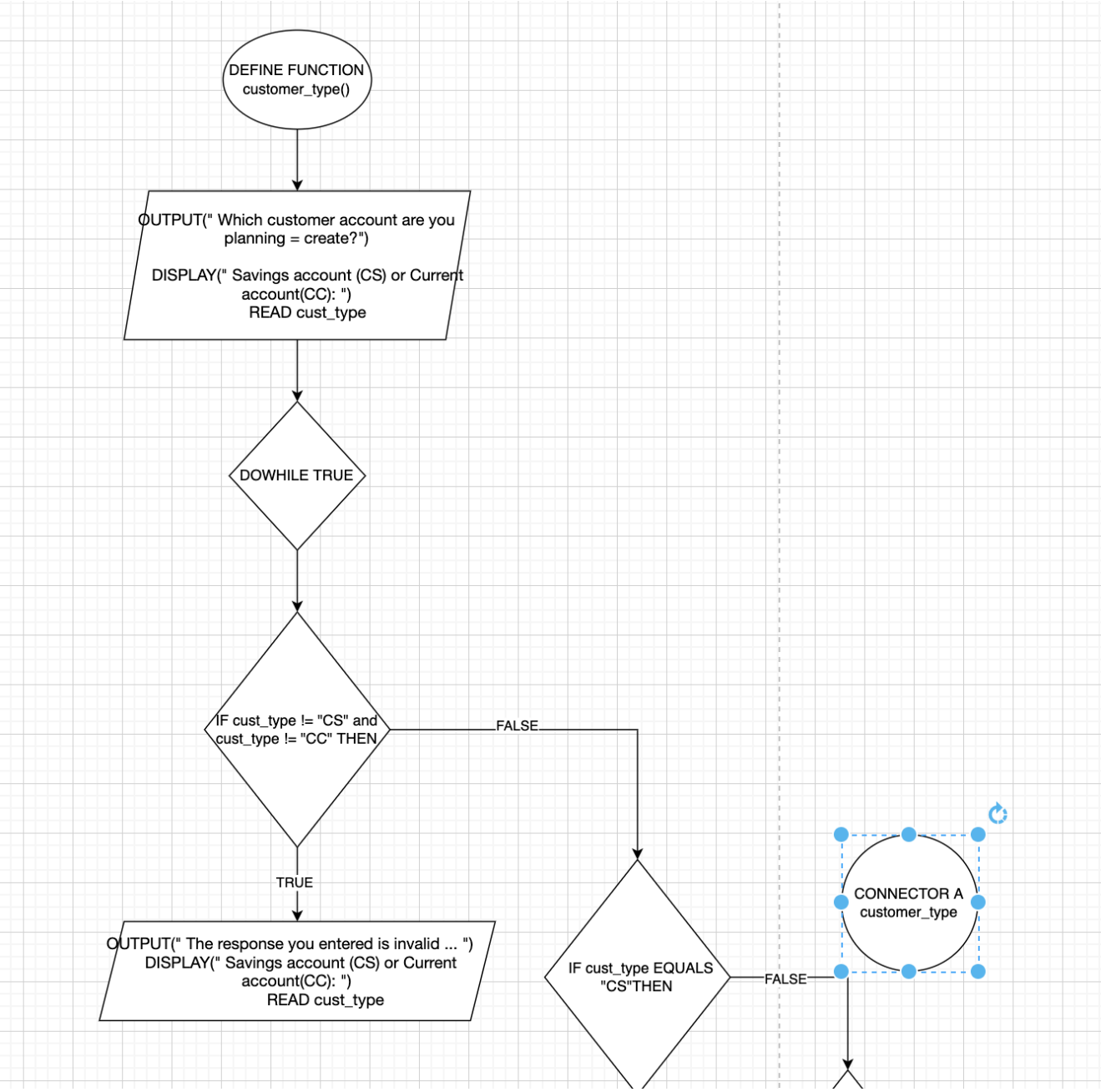

with open("data.txt", "a") as data :
OPEN ("data.txt") with append mode as DATA
data.write(registerdata)
OUTPUT(" Account has been created ID = " + new_id + " Password = " + new_pass)
OUTPUT(" Name : " + name + " Phone Number : " + number + " Email Address : " + email)
menu(userData,lineNumber)
CONNECTOR B create_id
END DEFINE

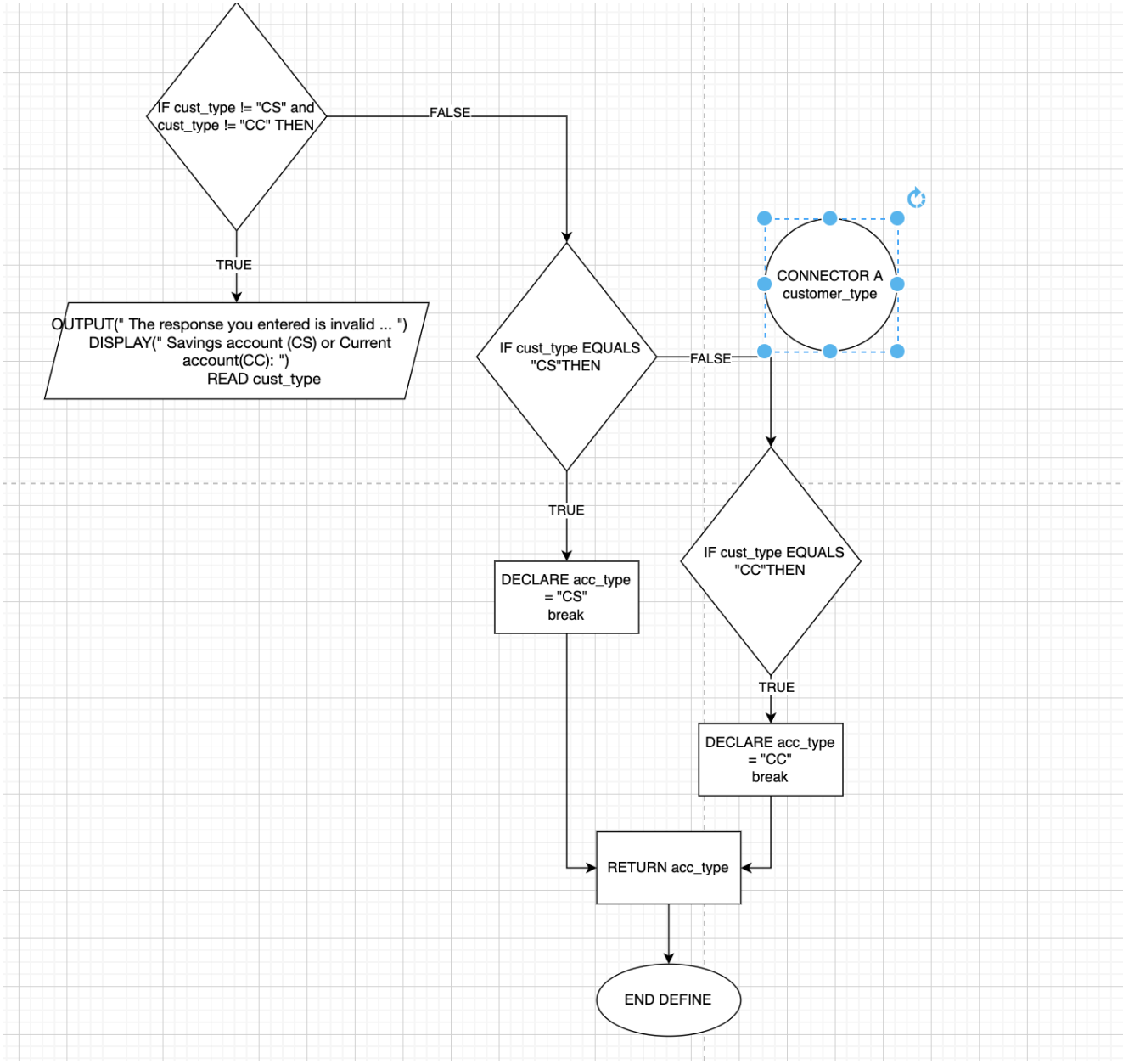
NEW USER DATA REGISTRATION



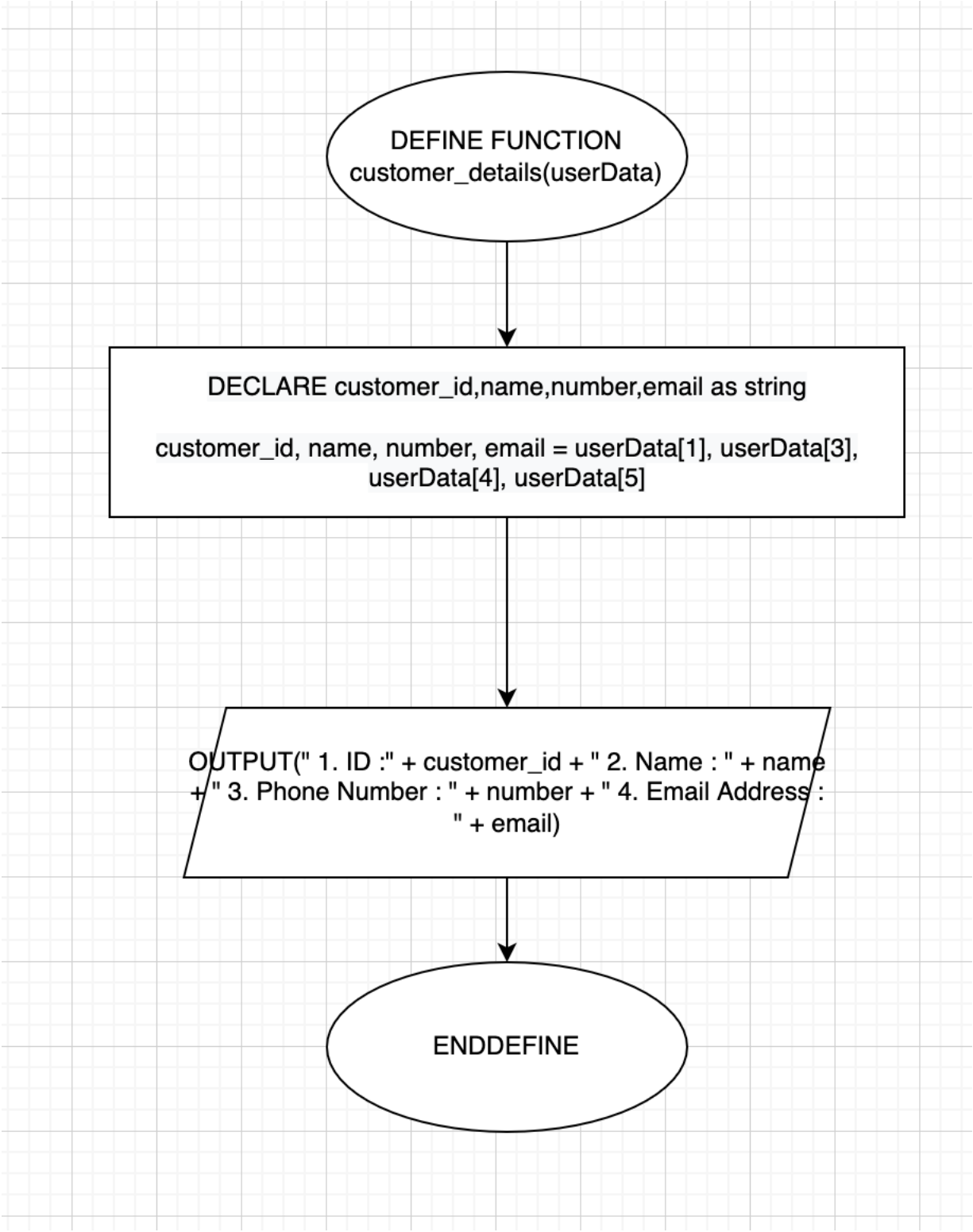

DISPLAY(" Enter Account owner' email address: ")
READ email
CONNECTOR A
userData_create
input_check(email,5)
create_id(name, number, email, acc_type, x, menu, userData, lineNumber)
main()
END DEFINE

CHECING CUSTOMER TYPE

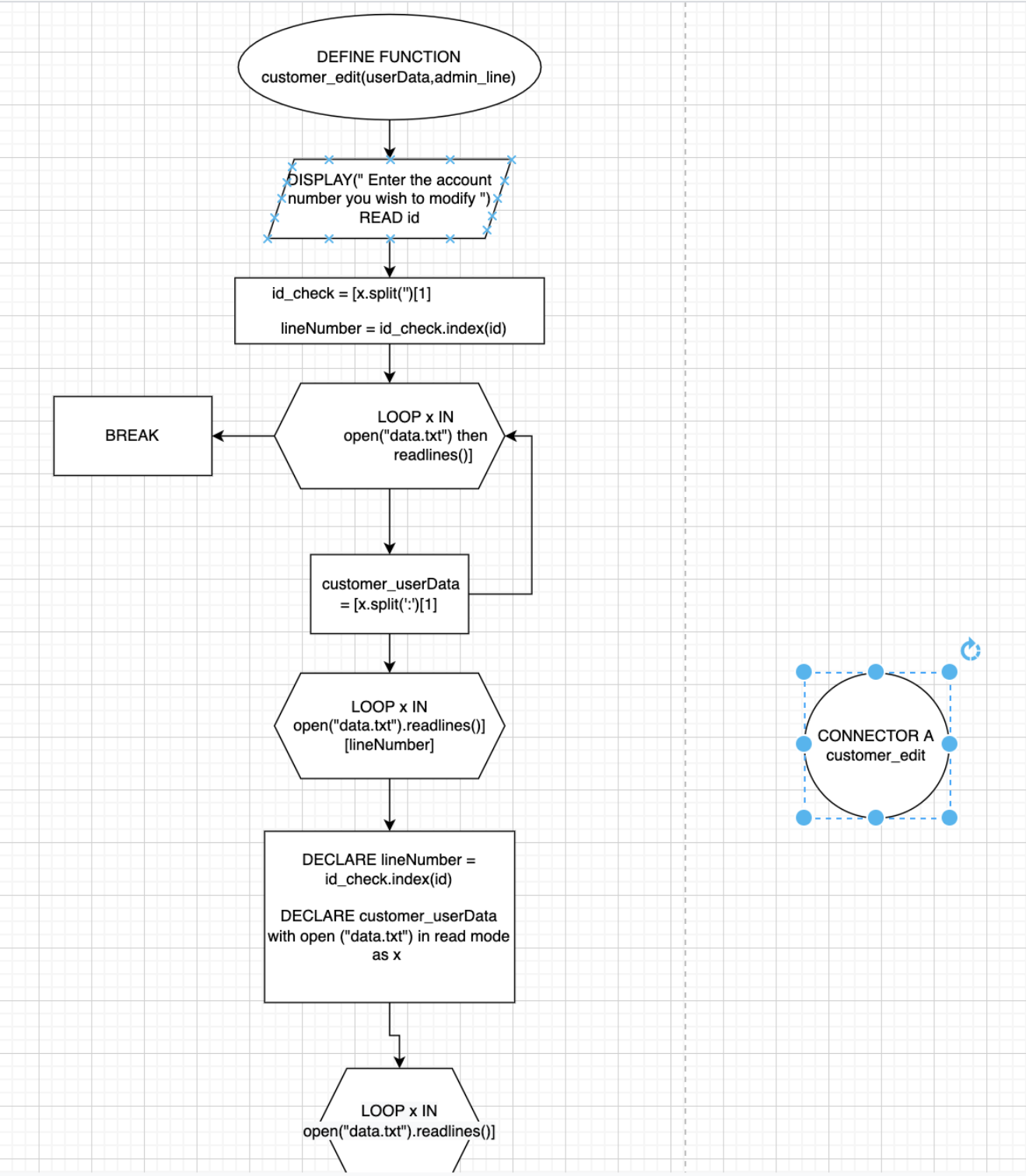


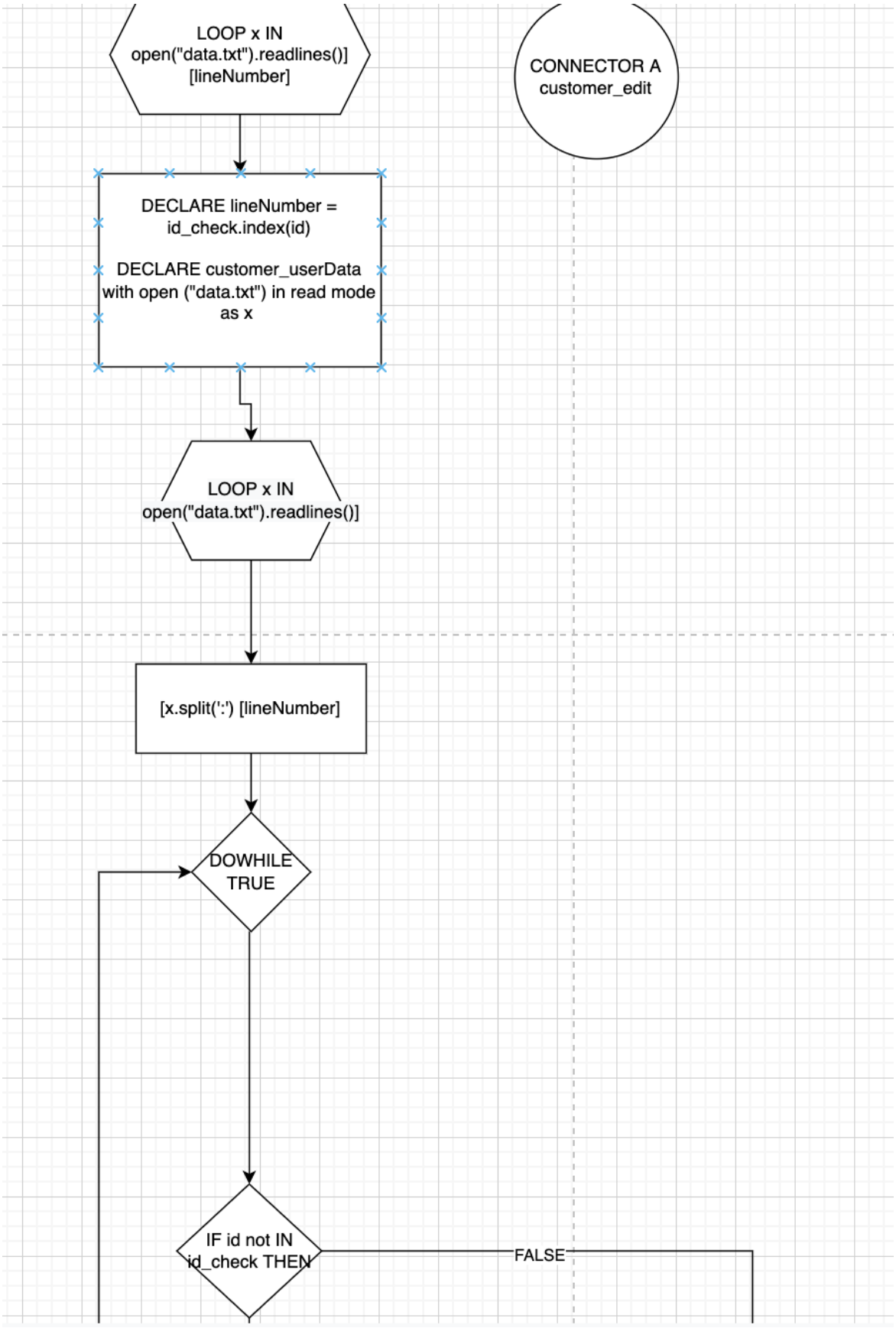

IF cust_type != "CS" and cust_type != "CC" THEN
FALSE
TRUE
OUTPUT(" The response you entered is invalid ... ")
DISPLAY(" Savings account (CS) or Current account(CC): ")
READ cust_type
IF cust_type EQUALS "CS"THEN
FALSE
CONNECTOR A
customer_type
TRUE
DECLARE acc_type = "CS"
break
IF cust_type EQUALS "CC"THEN
TRUE
DECLARE acc_type = "CC"
break
RETURN acc_type
END DEFINE

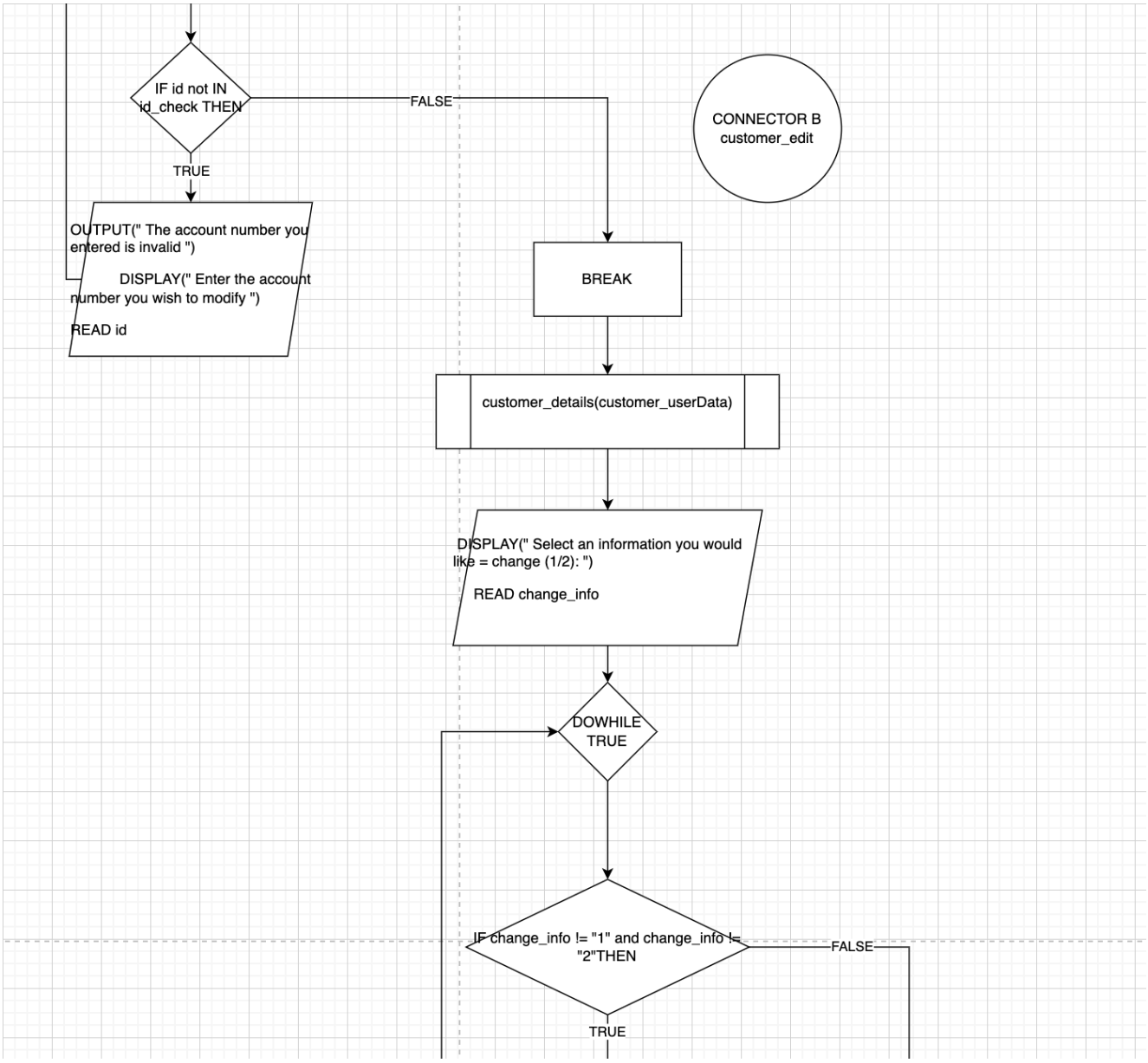
SHOW CUSTOMER DETAILS

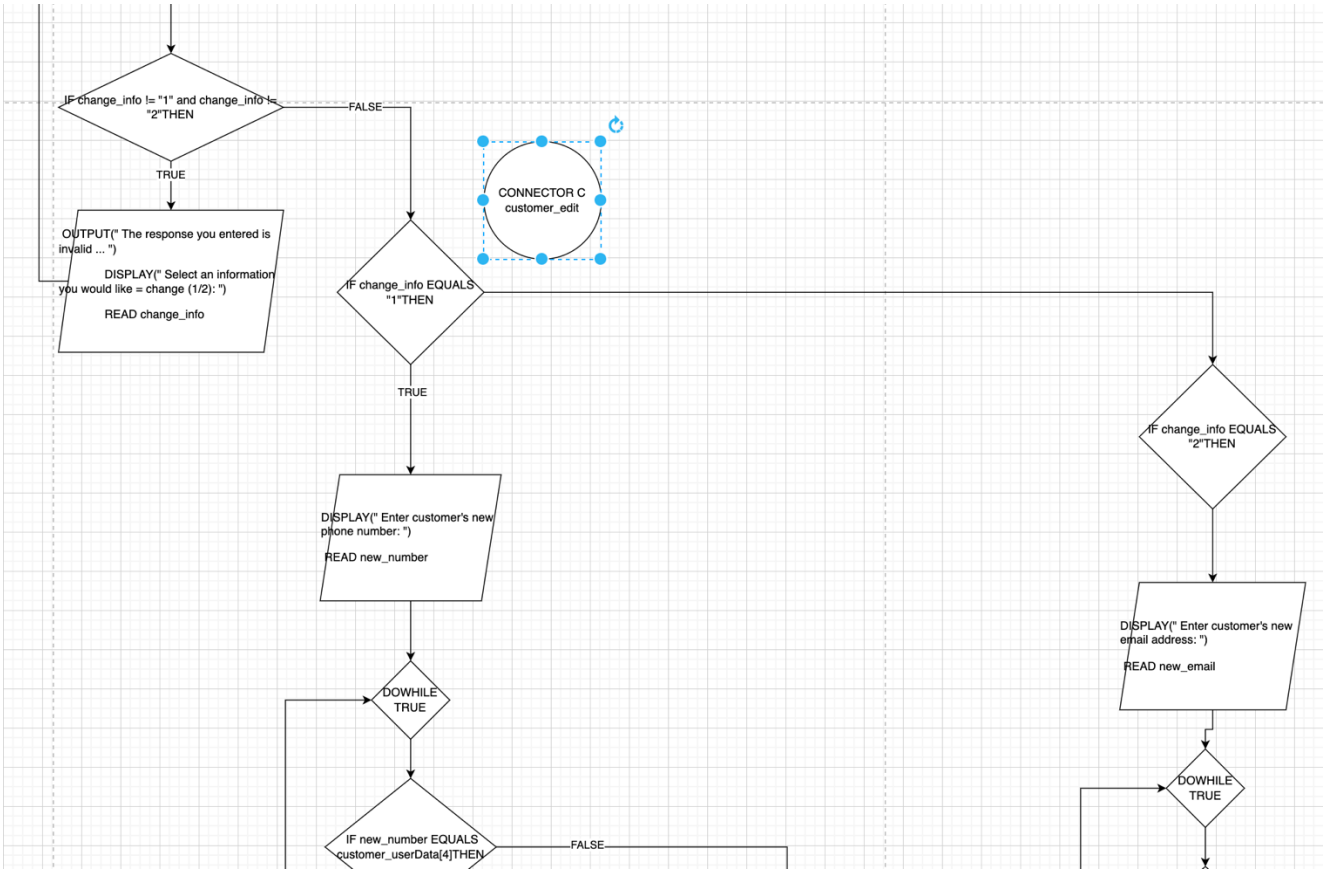


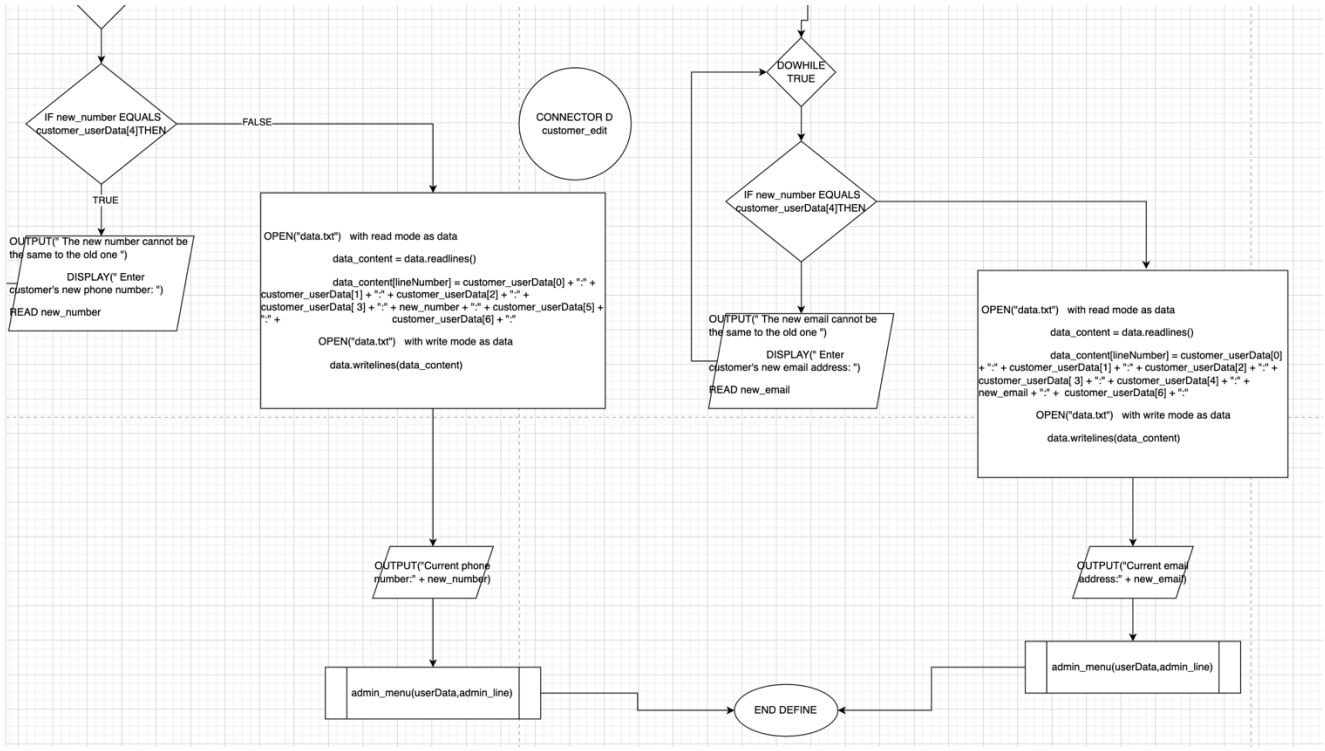
MODIFY CUSTOMER DATA



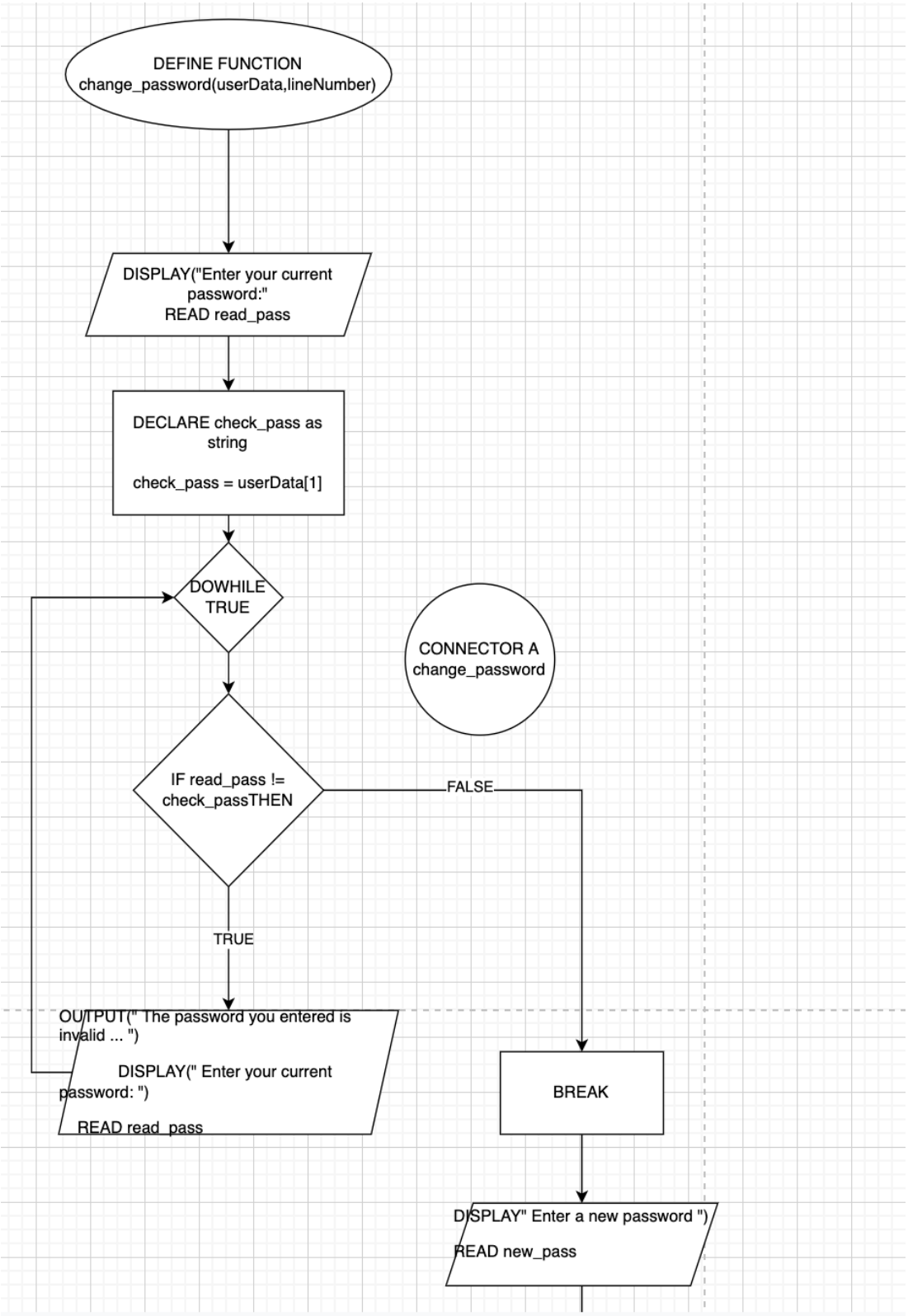

LOOP x IN
open("data.txt").readlines()]
[lineNumber]
CONNECTOR A
customer_edit
DECLARE lineNumber =
id_check.index(id)
DECLARE customer_userData
with open ("data.txt") in read mode
as x
LOOP x IN
open("data.txt").readlines()]
[x.split(':') [lineNumber]
DOWHILE
TRUE
IF id not IN
id_check THEN
FALSE

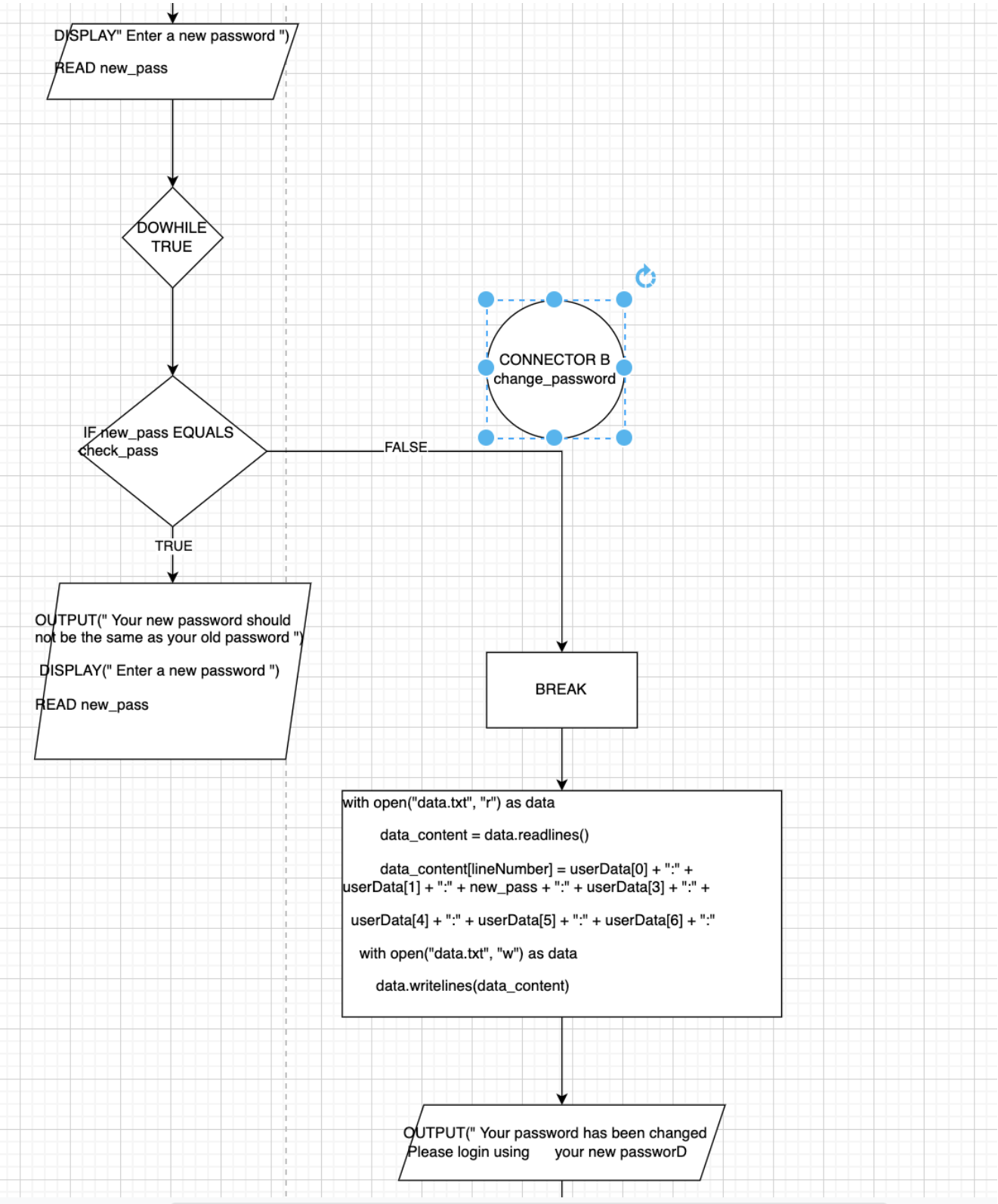

IF id not IN id_check THEN
FALSE
TRUE
CONNECTOR B customer_edit
OUTPUT(" The account number you entered is invalid ")
DISPLAY(" Enter the account number you wish to modify ")
READ id
BREAK
customer_details(customer_userData)
DISPLAY(" Select an information you would like = change (1/2): ")
READ change_info
DOWHILE TRUE
IF change_info != "1" and change_info != "2"THEN
FALSE
TRUE

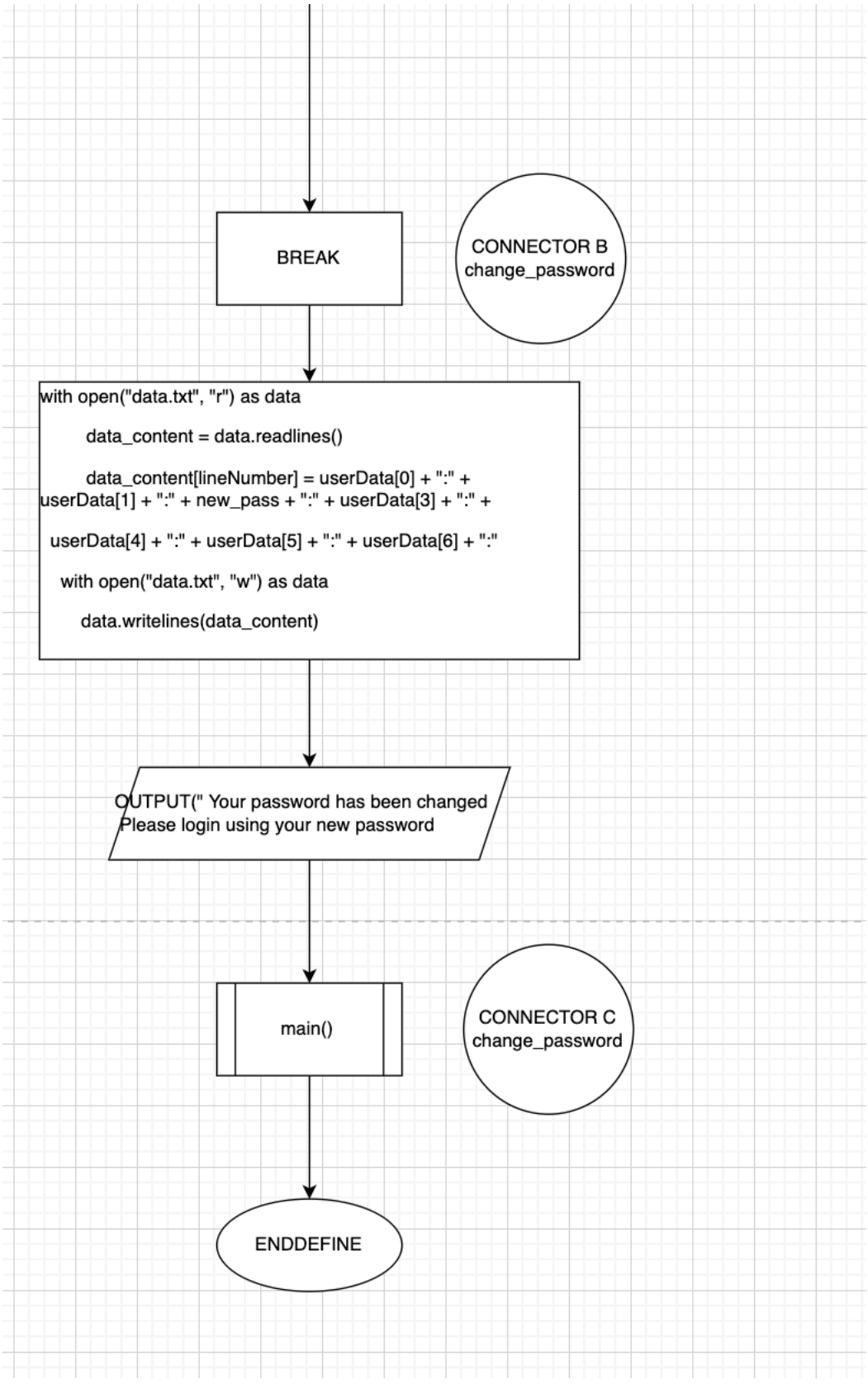

IF change_info != "1" and change_info != "2"THEN
FALSE
TRUE
OUTPUT(" The response you entered is invalid ... ")
DISPLAY(" Select an information you would like = change (1/2): ")
READ change_info
CONNECTOR C
customer_edit
IF change_info EQUALS "1"THEN
TRUE
DISPLAY(" Enter customer's new phone number: ")
READ new_number
DOWHILE TRUE
IF new_number EQUALS customer_userData[4]THEN
FALSE
IF change_info EQUALS "2"THEN
DISPLAY(" Enter customer's new email address: ")
READ new_email
DOWHILE TRUE


IF new_number EQUALS customer_userData[4]THEN
FALSE
TRUE
OUTPUT(" The new number cannot be the same to the old one ")
DISPLAY(" Enter customer's new phone number: ")
READ new_number
OPEN("data.txt") with read mode as data
data_content = data.readlines()
data_content[lineNumber] = customer_userData[0] + ":" + customer_userData[1] + ":" + customer_userData[2] + ":" + customer_userData[3] + ":" + new_number + ":" + customer_userData[5] + ":" + customer_userData[6] + ":"
OPEN("data.txt") with write mode as data
data.writelines(data_content)
CONNECTOR D
customer_edit
DOWHILE TRUE
IF new_number EQUALS customer_userData[4]THEN
OUTPUT(" The new email cannot be the same to the old one ")
DISPLAY(" Enter customer's new email address: ")
READ new_email
OPEN("data.txt") with read mode as data
data_content = data.readlines()
data_content[lineNumber] = customer_userData[0] + ":" + customer_userData[1] + ":" + customer_userData[2] + ":" + customer_userData[3] + ":" + customer_userData[4] + ":" + new_email + ":" + customer_userData[6] + ":"
OPEN("data.txt") with write mode as data
data.writelines(data_content)
OUTPUT("Current phone number:" + new_number)
OUTPUT("Current email address:" + new_email)
admin_menu(userData,admin_line)
admin_menu(userData,admin_line)
END DEFINE

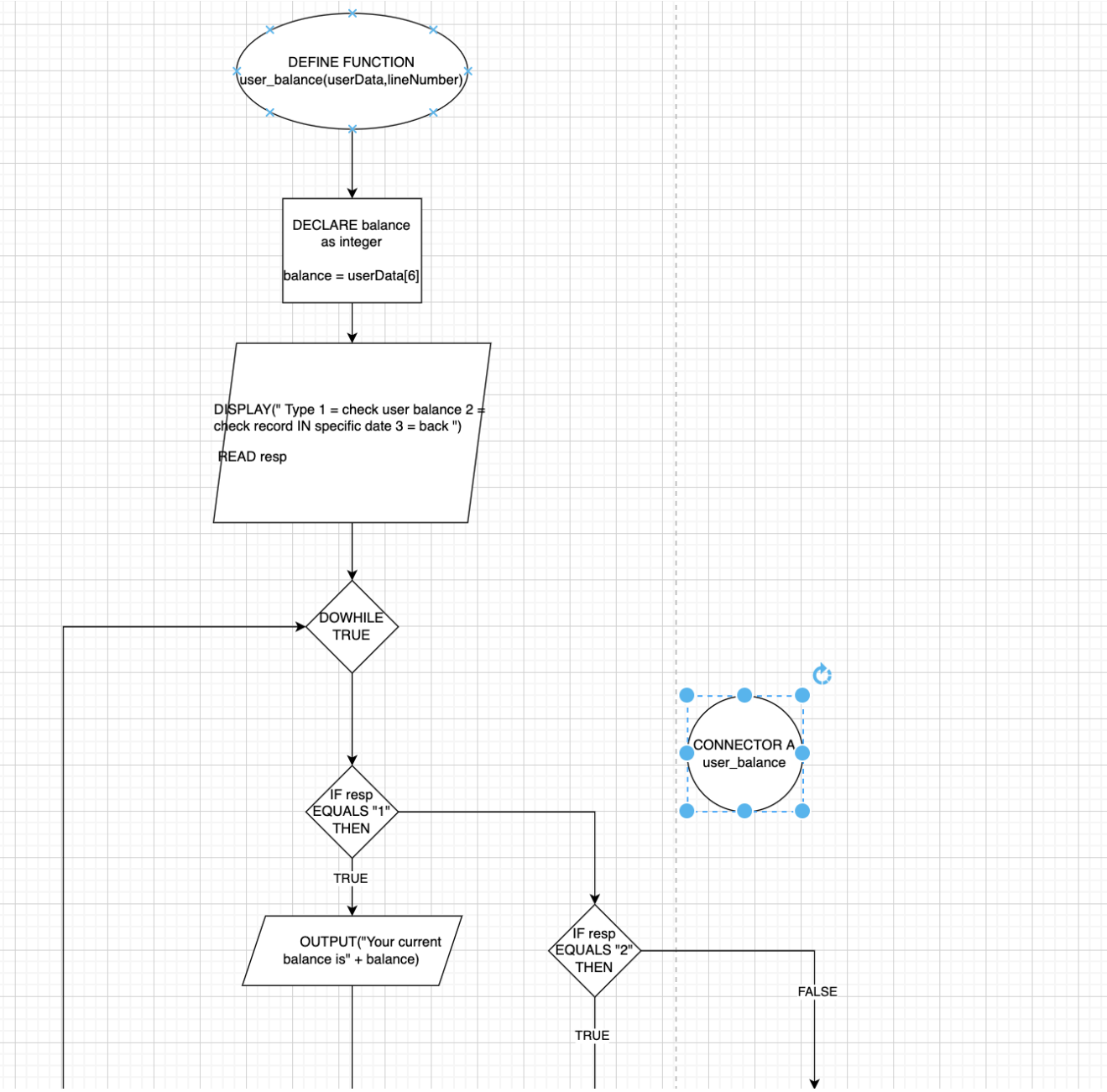
TO CHANGE PASSWORD

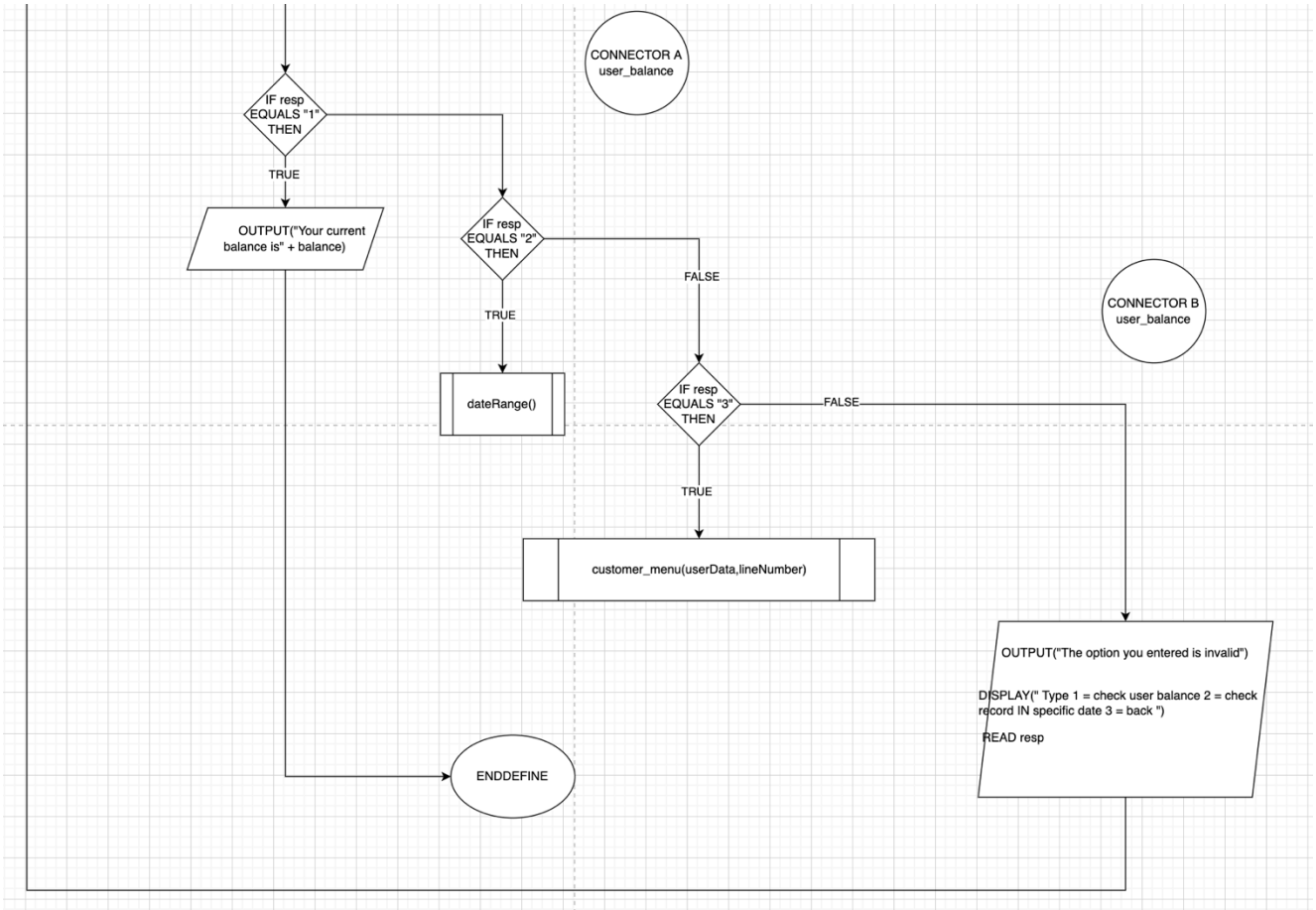



DISPLAY" Enter a new password ")
READ new_pass
DOWHILE TRUE
CONNECTOR B
change_password
IF new_pass EQUALS check_pass
FALSE
TRUE
OUTPUT(" Your new password should not be the same as your old password ")
DISPLAY(" Enter a new password ")
READ new_pass
BREAK
with open("data.txt", "r") as data
data_content = data.readlines()
data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + new_pass + ":" + userData[3] + ":" + userData[4] + ":" + userData[5] + ":" + userData[6] + ":"
with open("data.txt", "w") as data
data.writelines(data_content)
OUTPUT(" Your password has been changed Please login using your new passworD

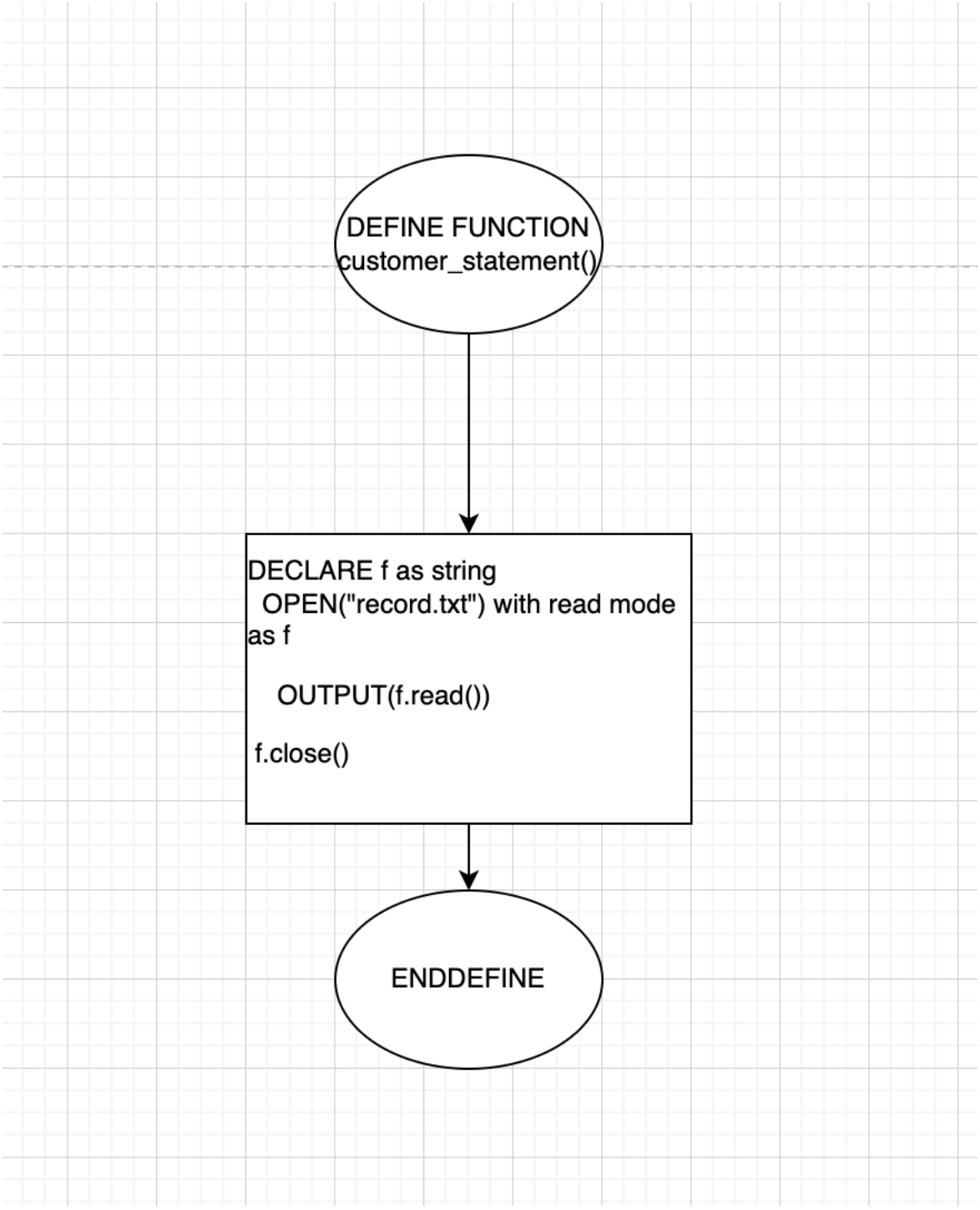

BREAK
CONNECTOR B
change_password
with open("data.txt", "r") as data
data_content = data.readlines()
data_content[lineNumber] = userData[0] + ":" +
userData[1] + ":" + new_pass + ":" + userData[3] + ":" +
userData[4] + ":" + userData[5] + ":" + userData[6] + ":"
with open("data.txt", "w") as data
data.writelines(data_content)
OUTPUT(" Your password has been changed
Please login using your new password
main()
CONNECTOR C
change_password
ENDDEFINE

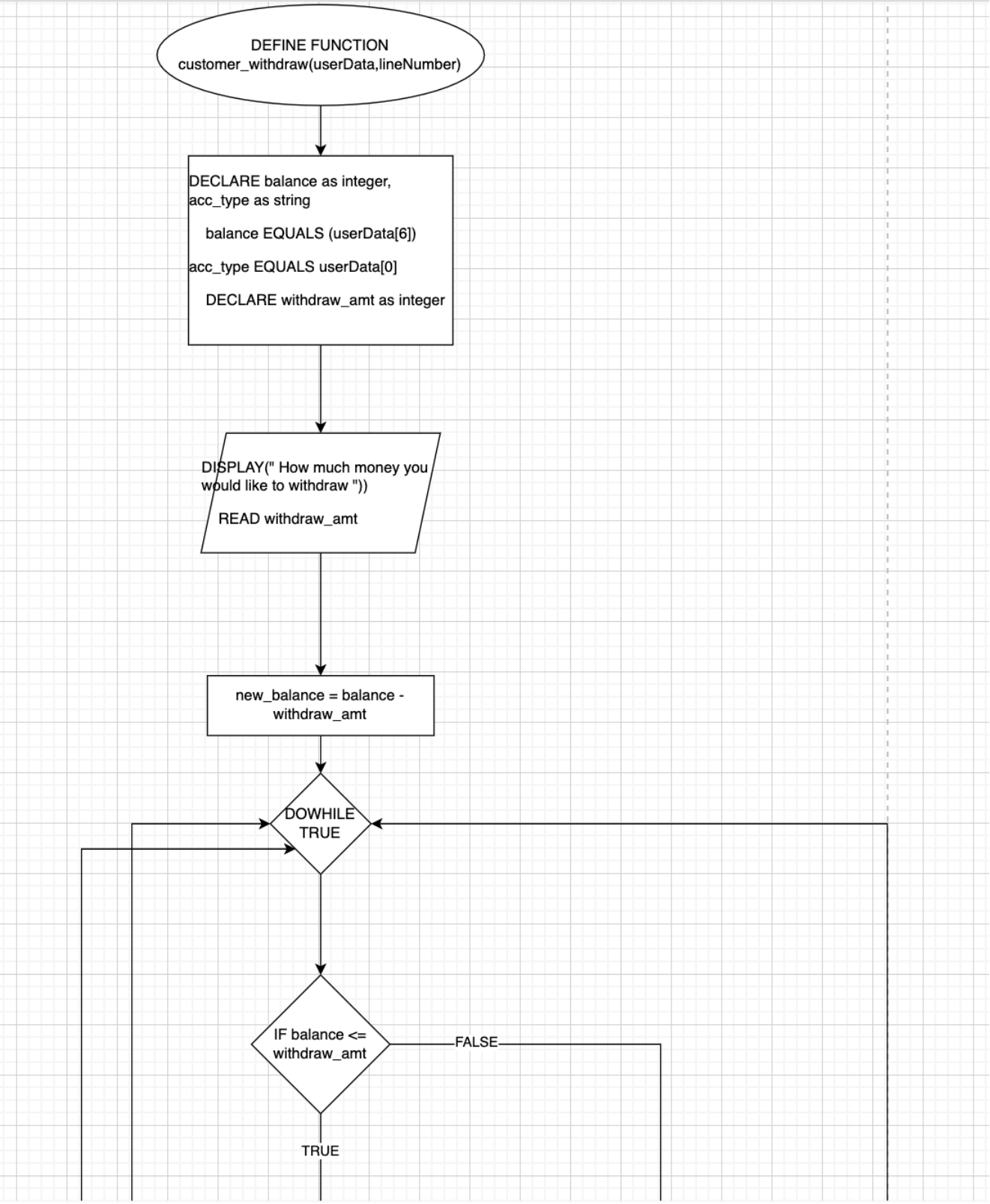
CHECKING USER BALANCE

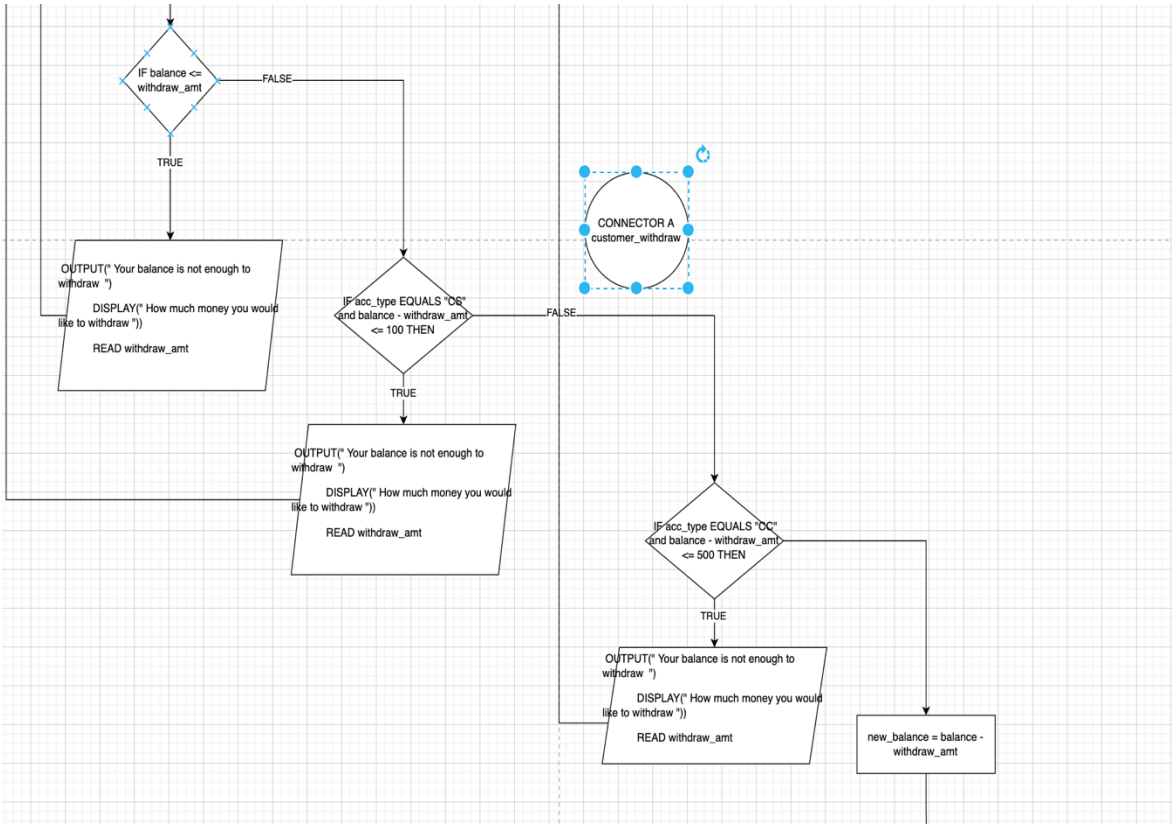


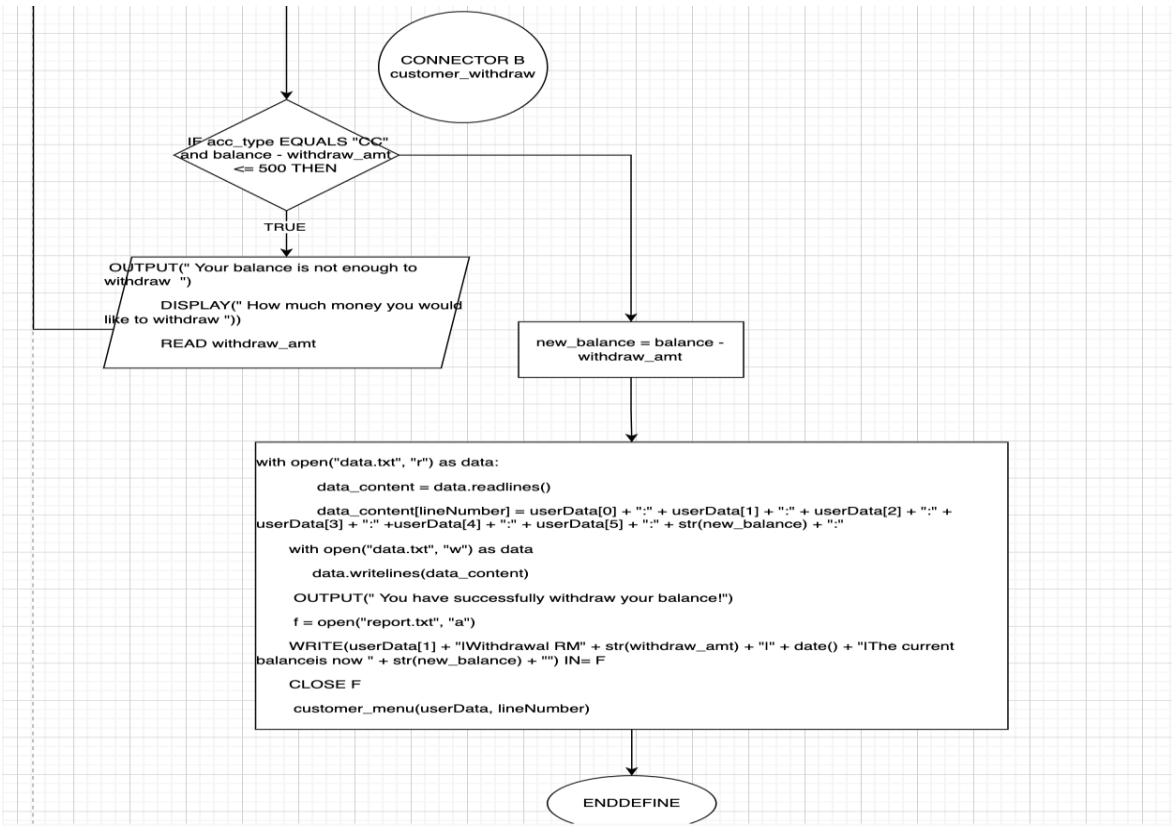

CONNECTOR A
user_balance
IF resp EQUALS "1" THEN
TRUE
OUTPUT("Your current balance is" + balance)
IF resp EQUALS "2" THEN
FALSE
TRUE
dateRange()
CONNECTOR B
user_balance
IF resp EQUALS "3" THEN
FALSE
TRUE
customer_menu(userData,lineNumber)
OUTPUT("The option you entered is invalid")
DISPLAY(" Type 1 = check user balance 2 = check record IN specific date 3 = back ")
READ resp
ENDDEFINE

ACCOUNT REPORT STATEMENT

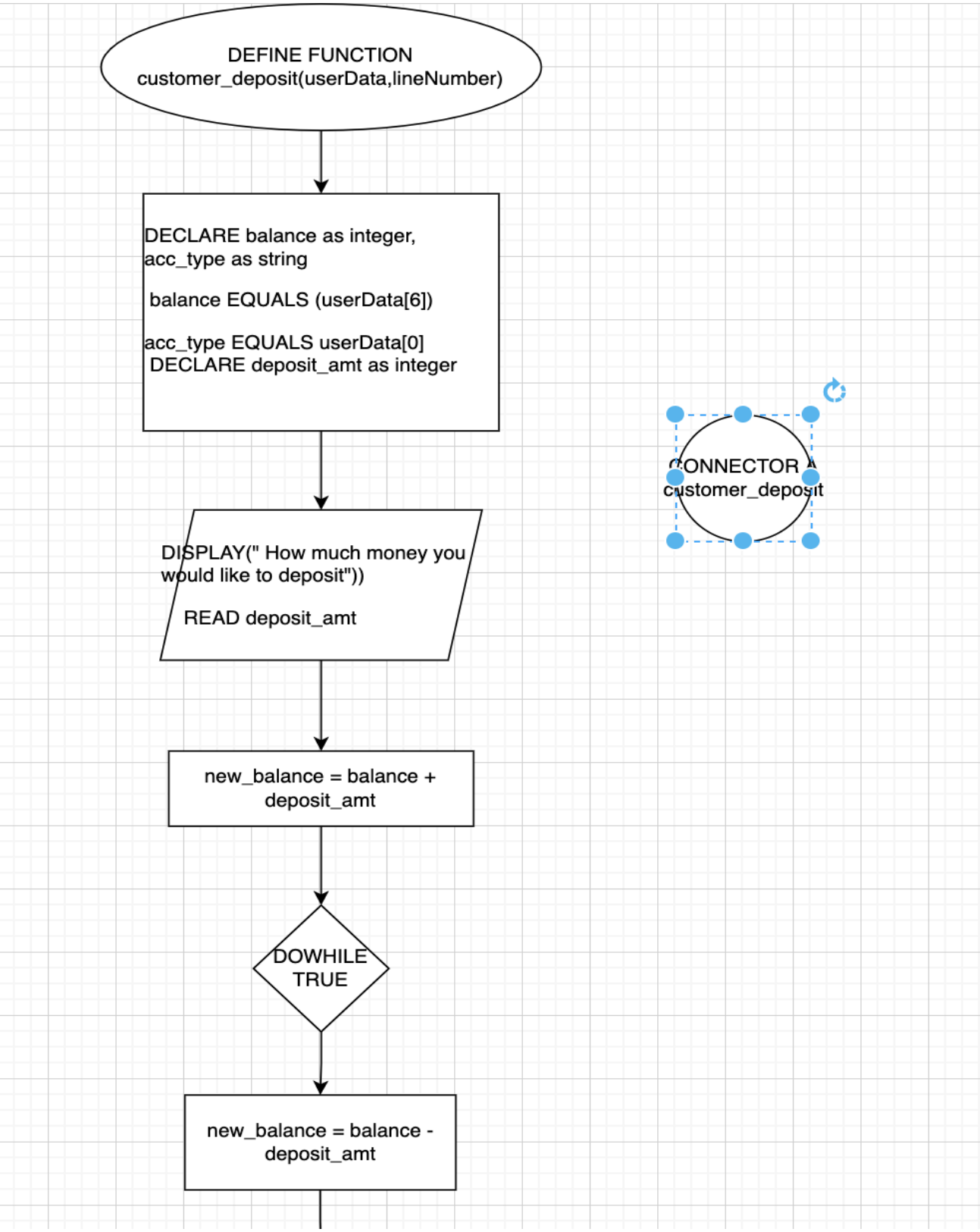


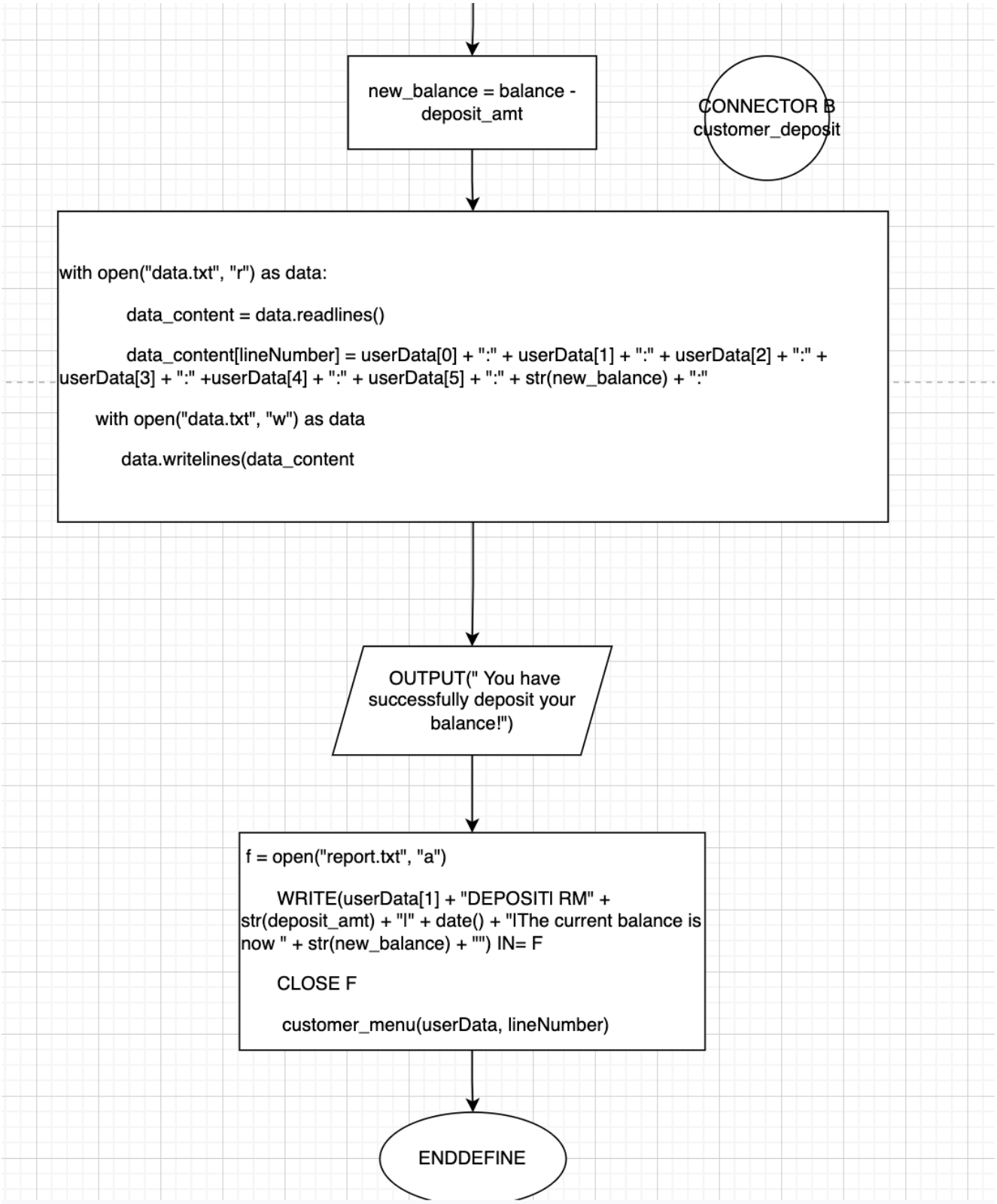

DEFINE FUNCTION
customer_withdraw(userData,lineNumber)
DECLARE balance as integer,
acc_type as string
balance EQUALS (userData[6])
acc_type EQUALS userData[0]
DECLARE withdraw_amt as integer
DISPLAY(" How much money you would like to withdraw "))
READ withdraw_amt
new_balance = balance - withdraw_amt
DOWHILE TRUE
IF balance <= withdraw_amt
FALSE
TRUE


IF balance <= withdraw_amt
FALSE
TRUE
OUTPUT(" Your balance is not enough to withdraw ")
DISPLAY(" How much money you would like to withdraw "))
READ withdraw_amt
IF acc_type EQUALS "OC" and balance - withdraw_amt <= 100 THEN
FALSE
TRUE
OUTPUT(" Your balance is not enough to withdraw ")
DISPLAY(" How much money you would like to withdraw "))
READ withdraw_amt
CONNECTOR A customer_withdraw
IF acc_type EQUALS "CC" and balance - withdraw_amt <= 500 THEN
TRUE
OUTPUT(" Your balance is not enough to withdraw ")
DISPLAY(" How much money you would like to withdraw "))
READ withdraw_amt
new_balance = balance - withdraw_amt

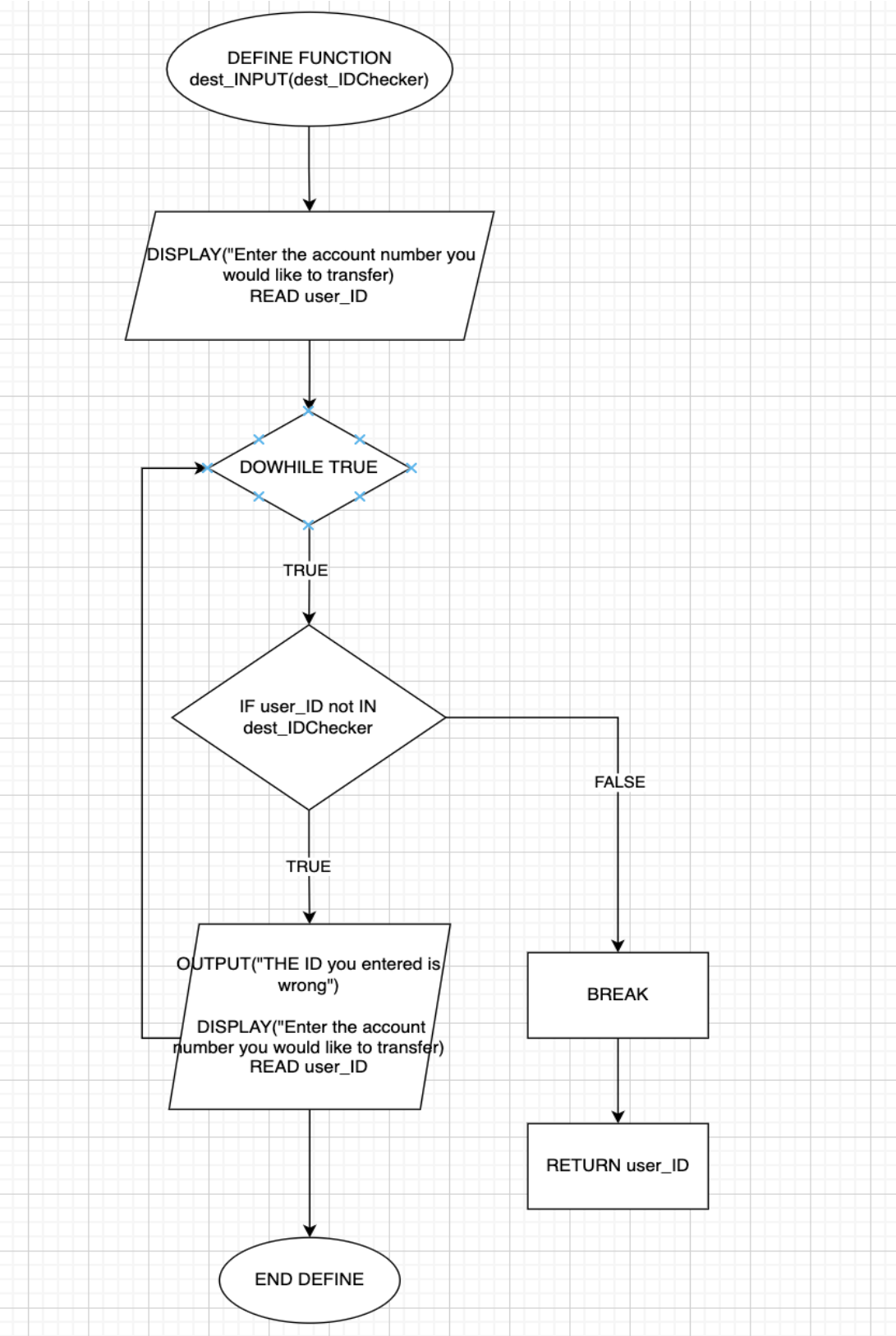

CONNECTOR B customer_withdraw
IF acc_type EQUALS "CC" and balance - withdraw_amt <= 500 THEN
TRUE
OUTPUT(" Your balance is not enough to withdraw ")
DISPLAY(" How much money you would like to withdraw "))
READ withdraw_amt
new_balance = balance - withdraw_amt
with open("data.txt", "r") as data:
data_content = data.readlines()
data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + userData[2] + ":" + userData[3] + ":" +userData[4] + ":" + userData[5] + ":" + str(new_balance) + ":"
with open("data.txt", "w") as data
data.writelines(data_content)
OUTPUT(" You have successfully withdraw your balance!")
f = open("report.txt", "a")
WRITE(userData[1] + "IWithdrawal RM" + str(withdraw_amt) + "I" + date() + "IThe current balanceis now " + str(new_balance) + "") IN= F
CLOSE F
customer_menu(userData, lineNumber)
ENDDEFINE

PERFORM TRANSACTION DEPOSIT

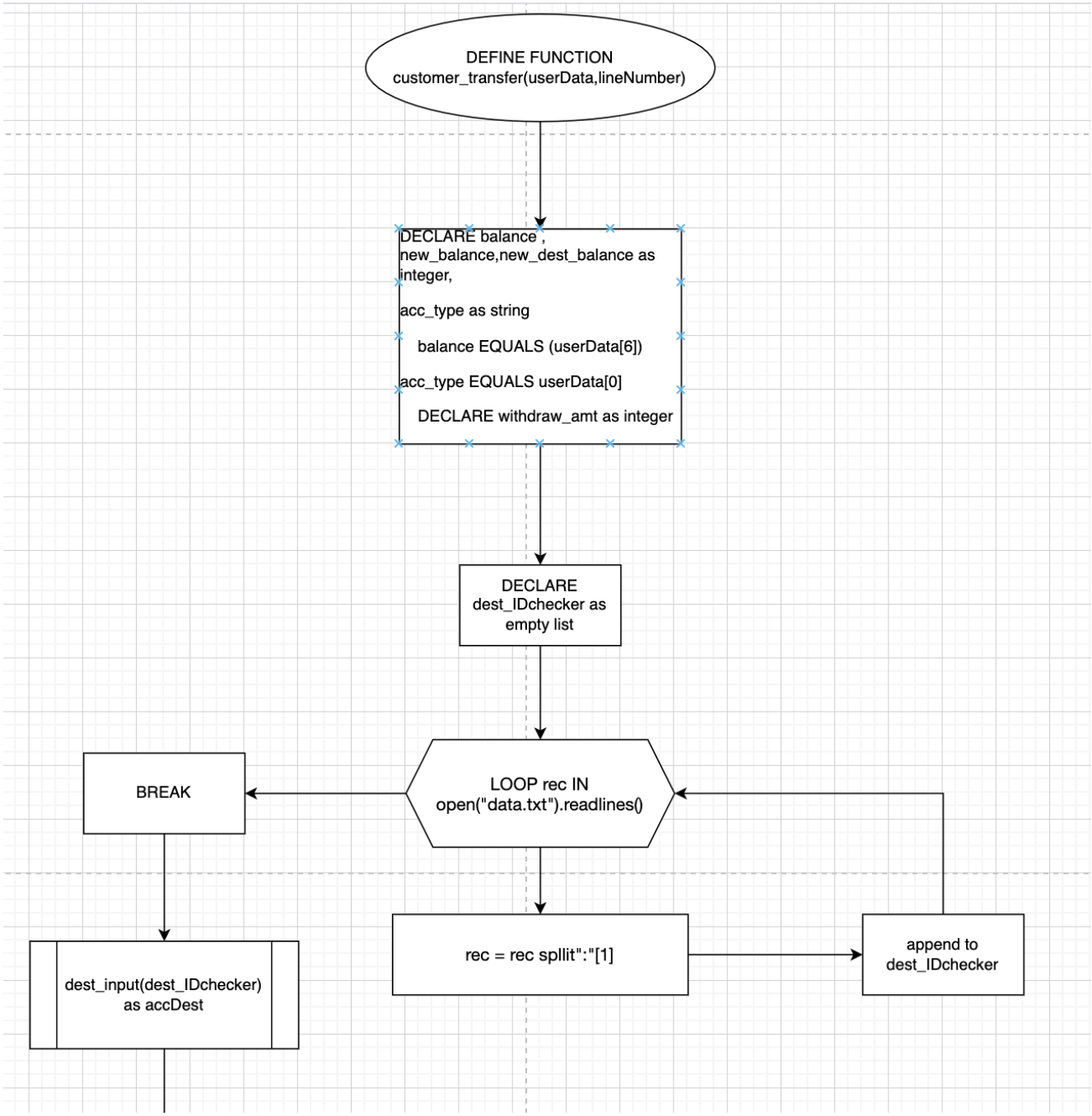


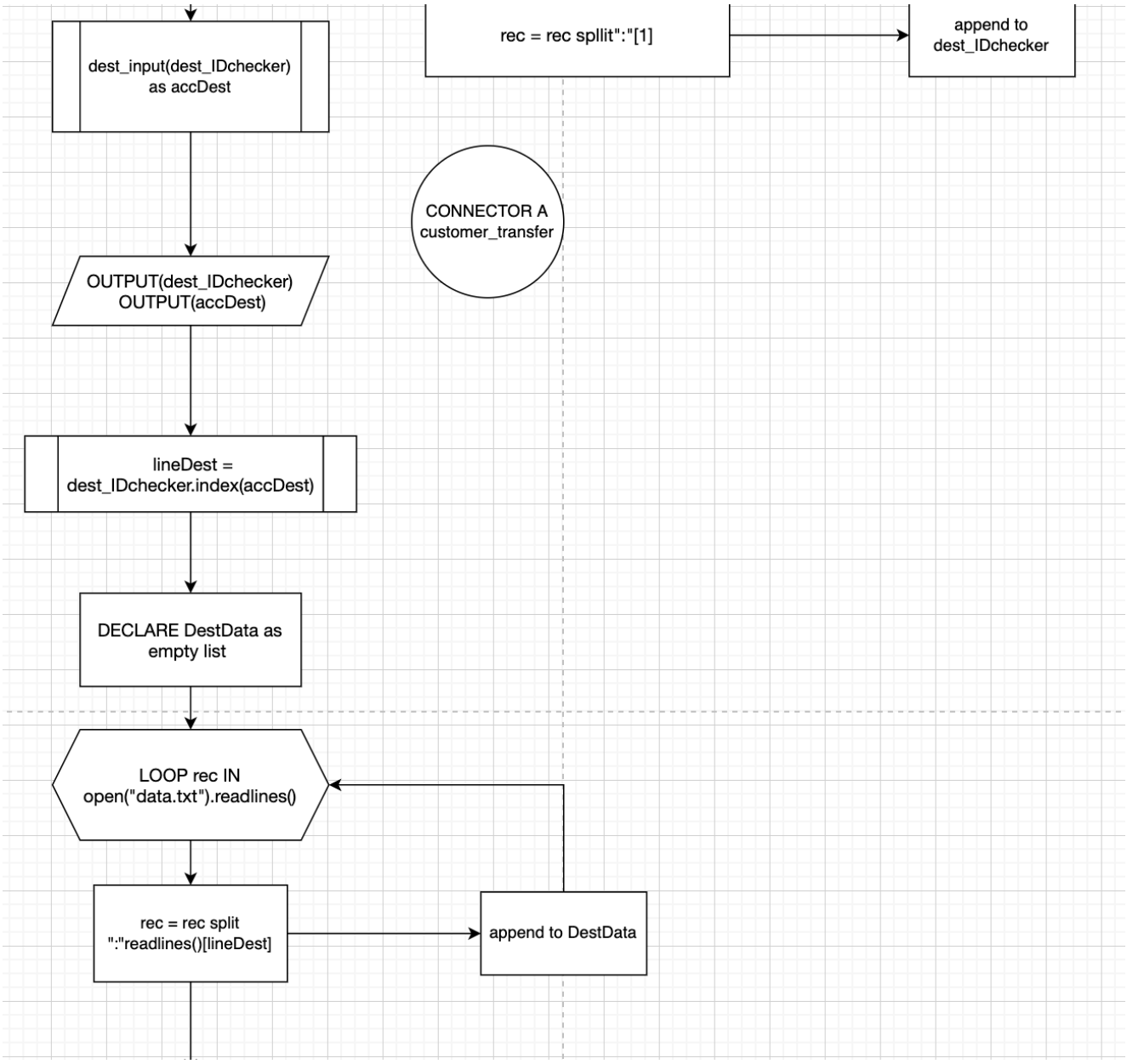

new_balance = balance - deposit_amt
CONNECTOR B customer_deposit
with open("data.txt", "r") as data:
data_content = data.readlines()
data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + userData[2] + ":" + userData[3] + ":" +userData[4] + ":" + userData[5] + ":" + str(new_balance) + ":"
with open("data.txt", "w") as data
data.writelines(data_content
OUTPUT(" You have successfully deposit your balance!")
f = open("report.txt", "a")
WRITE(userData[1] + "DEPOSITI RM" + str(deposit_amt) + "|" + date() + "|The current balance is now " + str(new_balance) + "") IN= F
CLOSE F
customer_menu(userData, lineNumber)
ENDDEFINE

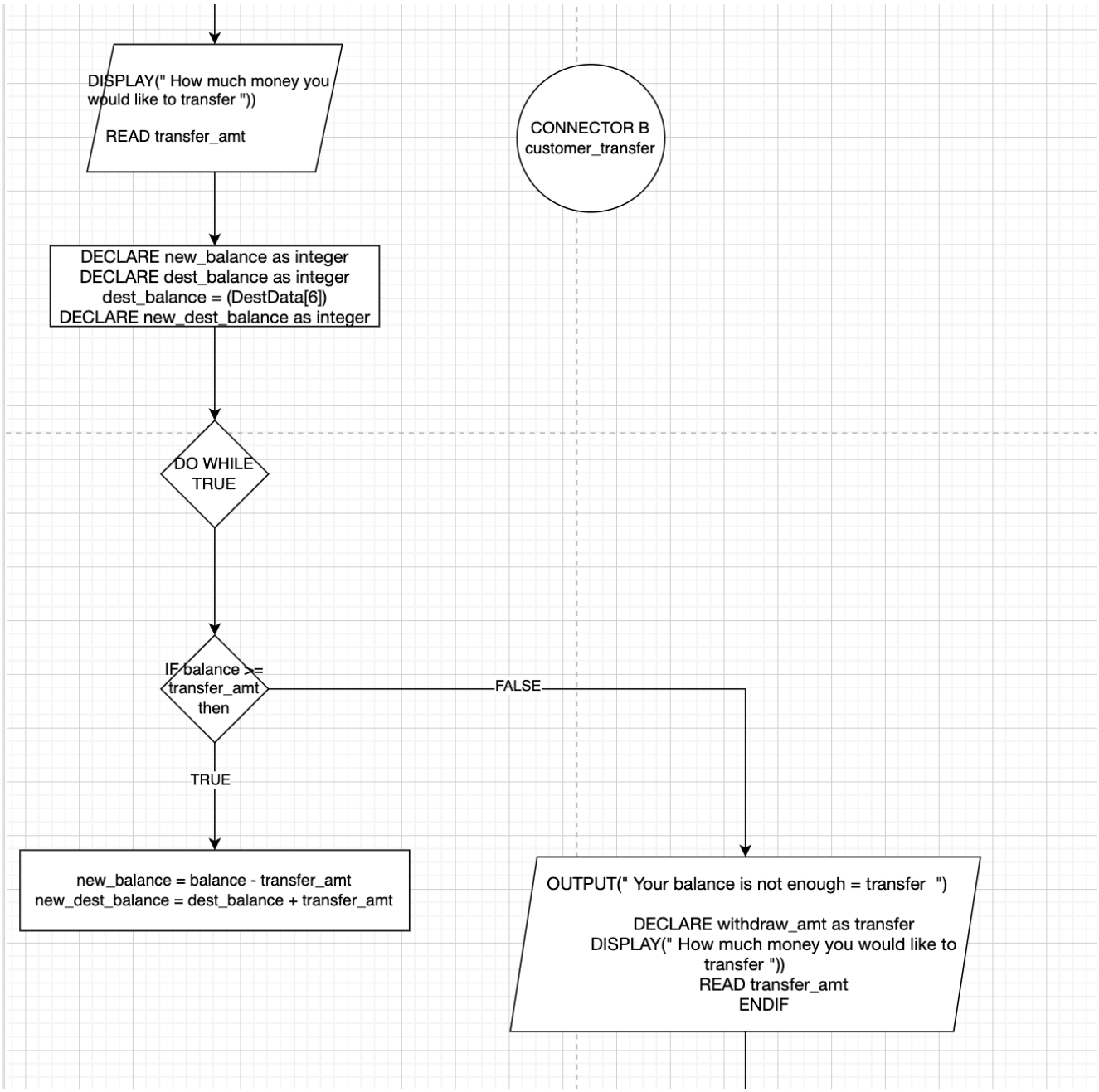
TO CHECK DESTINATION ID

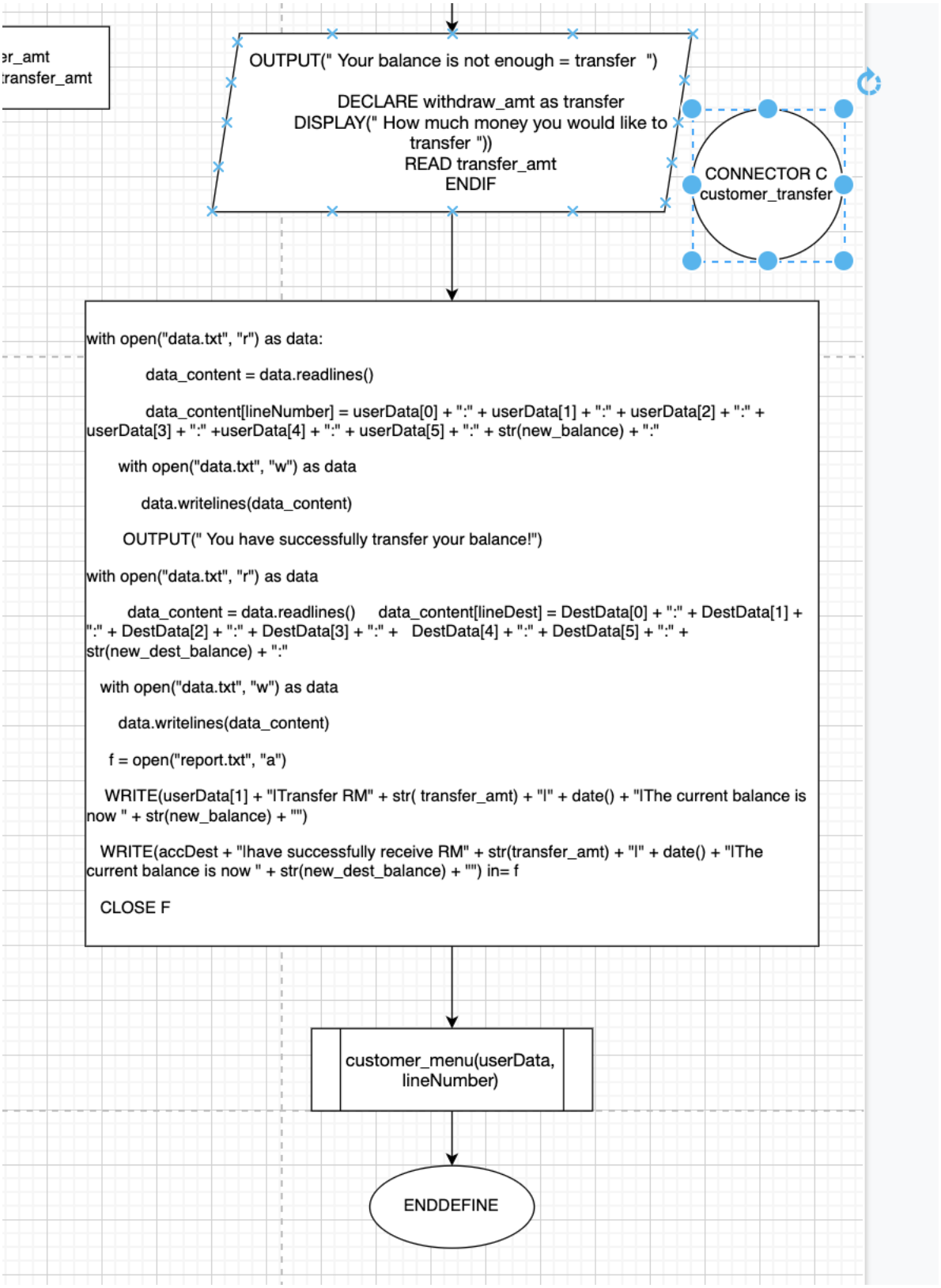


TO PERFORM TRANSACTION(TRANSFER)




rec = rec spllit":"[1]
append to dest_IDchecker
dest_input(dest_IDchecker) as accDest
CONNECTOR A customer_transfer
OUTPUT(dest_IDchecker) OUTPUT(accDest)
lineDest = dest_IDchecker.index(accDest)
DECLARE DestData as empty list
LOOP rec IN open("data.txt").readlines()
rec = rec split ":"readlines()[lineDest]
append to DestData


DISPLAY(" How much money you would like to transfer "))
READ transfer_amt
CONNECTOR B
customer_transfer
DECLARE new_balance as integer
DECLARE dest_balance as integer
dest_balance = (DestData[6])
DECLARE new_dest_balance as integer
DO WHILE TRUE
IF balance >= transfer_amt then
FALSE
TRUE
new_balance = balance - transfer_amt
new_dest_balance = dest_balance + transfer_amt
OUTPUT(" Your balance is not enough = transfer ")
DECLARE withdraw_amt as transfer
DISPLAY(" How much money you would like to transfer "))
READ transfer_amt
ENDIF


er_amt
transfer_amt
OUTPUT(" Your balance is not enough = transfer ")
DECLARE withdraw_amt as transfer
DISPLAY(" How much money you would like to transfer "))
READ transfer_amt
ENDIF
CONNECTOR C
customer_transfer
with open("data.txt", "r") as data:
data_content = data.readlines()
data_content[lineNumber] = userData[0] + ":" + userData[1] + ":" + userData[2] + ":" + userData[3] + ":" +userData[4] + ":" + userData[5] + ":" + str(new_balance) + ":"
with open("data.txt", "w") as data
data.writelines(data_content)
OUTPUT(" You have successfully transfer your balance!")
with open("data.txt", "r") as data
data_content = data.readlines() data_content[lineDest] = DestData[0] + ":" + DestData[1] + ":" + DestData[2] + ":" + DestData[3] + ":" + DestData[4] + ":" + DestData[5] + ":" + str(new_dest_balance) + ":"
with open("data.txt", "w") as data
data.writelines(data_content)
f = open("report.txt", "a")
WRITE(userData[1] + "lTransfer RM" + str(transfer_amt) + "l" + date() + "lThe current balance is now " + str(new_balance) + "")
WRITE(accDest + "lhave successfully receive RM" + str(transfer_amt) + "l" + date() + "lThe current balance is now " + str(new_dest_balance) + "") in= f
CLOSE F
customer_menu(userData, lineNumber)
ENDDEFINE

Program Source Code

Source code is the list of human-readable instructions that a programmer writes—often in a word processing program—when he is developing a program. The source code is run through a compiler to turn it into machine code, also called object code, that a computer can understand and execute (Thoughtco, 2019)

Variable

A variable is a fundamental concept in any programming language. It is a reserved memory location to store and manipulate values. In other words, a variable provides data to the computer to be processed. There are various datatypes that are used for the system, including string, integer, float, boolean and list. Any value like numbers or alphabets can be used to declare these variables such as 1, abc, fff, etc (Campbell, 2021)

String

```
user_ID = input("Enter your ID: ")
```

Figure 3.1.1.1

```
read_pass = input("Enter your password: ")
```

Figure 3.1.1.2

In *Figure 3.1.1.1*, the variable "user_ID" is assigned as a string datatype or coded as "str" to receive ID input from the user as an ID it usually contains characters and numbers. In *Figure 3.1.1.2*, the variable "read_pass" is also identified as a string datatype as they are being assigned using quotations.

Integer

```
balance = int(userData[6])
```

Figure 3.1.2.1

```
deposit_amt = int(input(" How much money you would like to deposit "))
```

Figure 3.1.2.2

In Figure 3.1.2.1 and Figure 3.1.2.2, the variable "balance" are first string and if we don't do casting, it won't be able to be combined with arithmetic operators like "+" , "-" , "*" , and etc. And integer must be numeric.

Boolean

```
if user_ID not in IDchecker:
    print("\nThe ID you entered is wrong\n")
    user_ID = input("Enter your ID: ")
else:
    break
```

Figure 3.1.3.1

```
if balance >= transfer_amt:
    new_balance = balance - transfer_amt
    new_dest_balance = dest_balance + transfer_amt
    break
else:
    print("\n Your balance is not enough to transfer  \n")
    transfer_amt = int(input(" How much money you would like to transfer "))
```

Figure 3.1.3.2

In an if statement there must be 2 possibilities either true or false, looking at figure 3.1.3.1 and 3.1.3.2 there can be 2 cases if it's true and if its false. In general, statement after the condition is usually considered as the "TRUE" one and statement after the "else" is the "FALSE" one.

List

```
IDchecker = []
```

Figure 3.1.4.1

```
userData = [rec.split(':') for rec in open("data.txt").readlines()][lineNumber]
```

Figure 3.1.4.2

In *Figure 3.1.5.1*, a declared empty list named as "IDchecker" which is used to store multiple data in a variable. While in "userData" it is also a list The "[linenumber]" is the number position of a list or is also known as index value.

Operator

Operators are used to perform operations on variables and values.(W3schools, 2019)

Arithmetic Operator

Substraction

```
new_balance = balance - withdraw_amt
```

Figure 3.2.1.1

Subtraction is also applied in the program. In *Figure 3.2.1.2*, it uses the operator "-" to subtract value "withdraw_amt" from "balance" resulting "new_balance".

Addition

```
new_balance = balance + deposit_amt
```

Figure 3.2.1.2

This code uses the operator "+" which is also knowns as the addition operator for adding together two values. The "new_balance" are sum of addition from "balance" and "deposit_amt"must be either integers or float data type in order to perform the action.

Multiplication

```
print("=" * 20)
```

Multiplication or "*" is an operator to add values to itself in certain times depends on the request. In this case print "=" for 20 times.

Assignment Operator

```
new_pass = str(int(existing_id[-1][-4:]) + 1)
```

Figure 3.2.2.1

In *Figure 3.2.2.1*, the variable "new_pass" is assigned to be the new password and it will add 1 each times new account is created

Comparison Operator

Equal and Not equal

```
if cust_response == "1":
```

Figure 3.2.3.1

```
if super_resp != "1" and super_resp != "2" and super_resp != "3":
```

Figure 3.2.3.2

In figure 3.2.3.1 The code will check whether the variable cust_response is equal to 1 or not. However in Figure 3.2.3.2 program can also be used to check if the two compared values are not equal or different values.

Less than, Less than or equal to

```
if balance <= withdraw_amt:
```

Figure 3.2.3.3

In figure 3.2.3.3

The code shown above works by checking as long as balance is less than or equal to withdraw_amt and it is assigned by "<=" equal sign after the less than sign

More than, More than or equal to

```
if balance >= transfer_amt:
```

Figure 3.2.3.4

In figure 3.2.3.4 the code process will check whether the balance is more than or equal to the transfer amount or not, it means that if the balance is less than the transfer amount or the same with the transfer amount process will be continued. It can be assigned by ">=" Equal sign after the more than sign.

Logical Operator

AND

```
elif acc_type == "CS" and balance - withdraw_amt <= 100 :
```

Figure 3.2.4.1

The figure 3.2.4.1 indicates that both condition/requirement must meet or in details, to continue the process acc_type must be equals to "CS" AND balance – withdraw amount must less than or equal to 100. However it cannot be only one of them that meet but it must both of them.

OR

```
elif check_type == "CC" or check_type == "CS":
```

Figure 3.2.4.2

There is also "OR" in logical operator, when "AND" means both condition must meet, "OR" is totally different. Whenever one of the condition is "TRUE" it will still proceed to the next part.

Membership Operator

Not In

```
if id not in id_check:
```

Figure 3.2.5.1

In figure 3.2.5.1 check whether the variable id can be found in variable id_check or not. If it cannot be found then the code results “TRUE” and the process will continue.

In

```
if input_item in check_input:
```

Figure 3.2.5.2

As shown in Figure 3.2.5.2 the code will check whether “input_item” can be found in the variable “check_input” or not. If it can be found, it generate “TRUE” value and the program will be executed

3.2.6 String Operator

Split

```
existing_id = [rec.split(':')[1] for rec in open("data.txt").readlines()]
```

Figure 3.2.6.1

The *Figure 3.2.6.1* provides a code “**split( )**” function for the purpose of splitting a string into a list. In the code, we have to choose a separator and put inside the function “:”. All string in the variable “rec” will be split into a list by separating every “:” and then assigned to the variable “existing_id”. For example rec contains “S:SUPER1000:1000:Mikael Owen:0817531422:mikaelowen2003@gmail.com:10000 , it will be read as “S,SUPER1000, 1000, Mikael Owen, 0817531422, mikaelowen2003@gmail.com , 10000 by using the function above.

Control Structure

If

```
if check_type == "S":
    super_menu(userData, lineNumber)
```

Figure 3.3.1

The single if statement is required to give condition, then it will decide whether the program will be executed or not. If the condition meet and the statement is true then the program below the if statement will be executed. On the other hand if the condition does not meet the requirement it will skip the If statement

If – Else

```
if user_ID not in IDchecker:
    print("\nThe ID you entered is wrong\n")
    user_ID = input("Enter your ID: ")
else:
    break
```

Figure 3.3.2

The IF ELSE statement is required to give condition, it is almost the same with the single IF statement, But in IF ELSE when the condition meet and the statement is true then the program below the if statement will be executed and if the condition does not meet the requirement it will execute the program below the ELSE statement.

If – Elif – Else

```
if cust_response == "1":
    customer_deposit(userData, lineNumber)
elif cust_response == "2":
    customer_withdraw(userData, lineNumber)
elif cust_response == "3":
    customer_transfer(userData, lineNumber)
elif cust_response == "4":
    user_balance(userData, lineNumber)
elif cust_response == "5":
    customer_statement(userData)
elif cust_response == "6":
    change_password(userData, lineNumber)
elif cust_response == "7":
    main()
else:
    print("\n The response you entered is invalid ... \n")
    break
```

Figure 3.3.3

The IF – ELIF – ELSE statement is like combination between the IF and IF ELSE statement. The different is just whenever the first IF does not meet the condition it will skip and continue the ELIF statement. After all ELIF statement check but still does not match with the condition. Lastly, it will execute the program below the ELSE.

Repetition Structure

While Loop

```
while True:
    if cust_type != "CS" and cust_type != "CC":
        print("\n The response you entered is invalid ... \n")
        cust_type = input(" Savings account (CS) or Current account(CC): ")
    elif cust_type == "CS":
        acc_type = "CS"
        break
    elif cust_type == "CC":
        acc_type = "CC"
        break
return acc_type
```

Figure 3.4.1

The While Loop, according to the name we can execute the statements as long as the condition is TRUE over and over again. After the condition become FALSE it will stop the loop and continue to the next command , Therefore remember to input limit or to increment or else the loop will remain forever.

For Loop

```
for rec in open("report.txt").readlines():
```

Figure 3.4.2

However, the for loop offer us something different but with the similar purpose. It is used for iterating over a sequence, either it is a list,string, integer, tuple, and etcIn *Figure 3.4.2.1*, it states that for every record stored in a text file will be executed by the following indented command. The For Loop structure will stop looping until all records have been read by the system without the need of any incrementation.

List

```
IDchecker.append(rec.split(":")[1])
```

Figure 3.5

In Figure 3.5.1, the purpose of is checking customer ID with the text files. Customer's data stored inside the text file were split into a list by the "split(:)" function and then picking up the correct index from the right customer list, which uses "[1]" The numbers inside "[]" are the index position of a list.

Function

```
def create_id(name, number, email, acc_type, rec, menu, userData, lineNumber):
    existing_id = [rec.split(':')[1] for rec in open("data.txt").readlines()]
    new_pass = str(int(existing_id[-1][-4:]) + 1)
    if acc_type == "CS":
        new_id = "SAVINGS" + new_pass
        registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ":" + "100" + "\n"
    elif acc_type == "CC":
        new_id = "CURRENT" + new_pass
        registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + ":" + "500" + "\n"
    else:
        new_id = "ADMIN" + new_pass
        registerdata = acc_type + ":" + new_id + ":" + new_pass + ":" + name + ":" + number + ":" + email + "\n"
    with open("data.txt", "a") as data:
        data.write(registerdata)
    print("\n Account has been created\n\n ID = " + new_id + "\n Password = " + new_pass)
    print(" Name : " + name + "\n Phone Number : " + number + "\n Email Address : " + email)
    menu(userData, lineNumber)
```

Figure 3.6.1

```
def userData_create(x, acc_type, menu, userData, lineNumber):
    print("\n  CREATE A NEW " + x + " ACCOUNT")
    print("=" * 20)
    name = input(" Enter Account owner's name: ")
    input_check(name, 3)
    number = input("\n Enter Account owner's phone number: ")
    input_check(number, 4)
    email = input("\n Enter Account owner's email address: ")
    input_check(email, 5)
    create_id(name, number, email, acc_type, x, menu, userData, lineNumber)
    main()
```

Figure 3.6.2

Python Functions is a block of related statements designed to perform a computational, logical, or evaluative task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.(GeeksForGeeks, 2021)

No	Screenshot	Function Name	Explanation
1		main	This function is where user login by inserting ID and password , usually called main menu.
2		id_input	This function works by matching whether the ID that the user inputted is match with the one in the text files or not
3		dest_input	Whenever user want to transfer some amount of money, they have to decide their destination. This is for checking whether the destination ID exists in the text files or not
4		pass_input	This function works by matching whether the password that the user inputted is

			match with the one in the text files or not
5	`def account_type(userData, lineNumber):`	account_type	Each account will proceed to different menu, this function is to categorized each account type to go to different menu.
6	`def super_menu(userData, lineNumber):`	super_menu	This function is a Super Menu, to generate,create a new Admin Account and change password
7	`def admin_menu(userData, lineNumber):`	admin_menu	This function is admin menu, where we can create customer account, edit customer's details, change password, and see customer statement of account report
8	`def customer_menu(userData, lineNumber):`	customer_menu	Customer menu function is where user can perform transaction such as deposit ,withdrawal and transfer. They can also check their balance, View account report, and change password.
9	`def input_check(input_item, index_num):`	input_check	This function is to make sure that there is no

			customer's data are the same (include name, email, phone number)
10		create_id	This function is to auto generate ID and password
11		userData_create	This function is to fill registration form for user data, including name, phone number, and email then send it to create_id function.
12		customer_type	Since customer account is divided into 2 types, this function is to split and categorize it.
13		customer_details	This function main purpose is to show full customer details.
14		customer_edit	Admin can edit customer data by using this function.
15		change_password	Every user can modify their password as they want. This function main purpose is to change password
16		user_balance	To show user balance and transaction in specific date.
17		customer_statement	To print customer's

			statement account report
18		customer_withdraw	To perform transaction (withdraw) we use this function. Then it will also modify our balance in the text files
19		customer_deposit	To perform transaction (deposit) we use this function. Then it will also modify our balance in the text files
20		customer_transfer	To perform transaction (transfer) we use this function. Then it will also modify our balance in the text files
21		date	To define date and import datetime library and also give format to the date, mine is Wednesday,1 January 2021
22		dateRange	To select specific date and print report between those two dates

File

Write

Figure 3.7.1

When we want to write data into a file we use the code “w”. The purpose of this write mode is to overwrite any previous data stored in the file to a new data. In *Figure 3.7.1*. In this case I use it to modify customer data.

Read

Figure 3.7.2

However, if we want to read data from a text files, then we have to use the code “r” . The file must be first “open” and assigned to a new variable.

Append

Figure 3.7.3

At last, append mode is almost the same with write mode but the append mode is more likely adding new data into the file and not overwrite it. Below the figure 3.7.3 there must be more statement of something that we would like to append.

Sample of Input and Output

Login menu

Figure 4.1.1

First the banking system will ask our ID , just like every site, applications as usual we have to login.

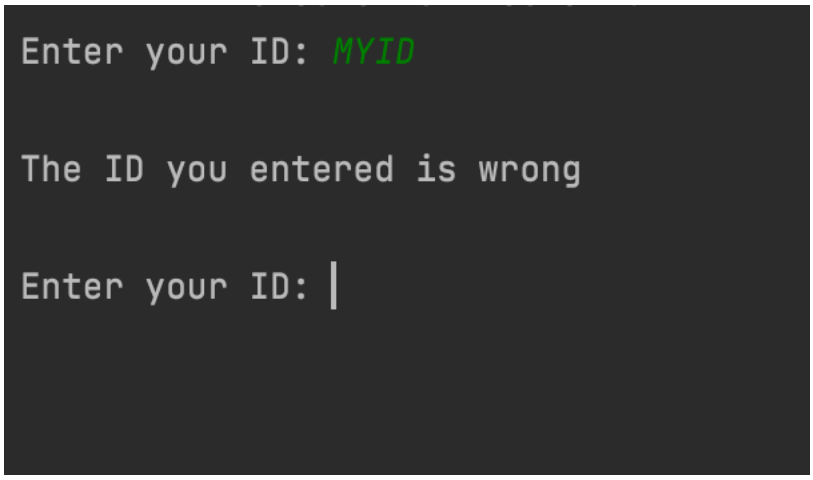


Figure 4.1.2

Then whenever we input wrong ID, the output will be like this and ask for more input.

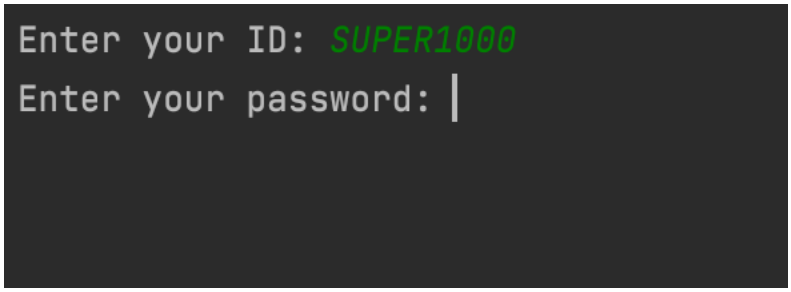


Figure 4.1.3

In figure 4.1.3 Until we finally we inputted the correct ID, the system will proceed and ask the next input. It is password.

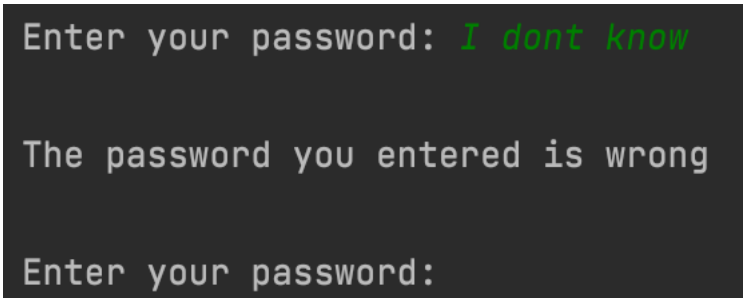


Figure 4.1.4

In figure 4.1.4,it is just the same with inputting ID, if it is wrong the system will ask for more input.

```
The password you entered is wrong

Enter your password: I dont know

The password you entered is wrong

Enter your password: 1000

====================Welcome to Super User Menu====================

 1. Create an admin account
 2. Change password
```

Figure 4.1.5

When we entered the correct password as in Figure 4.1.5 the system will continue, as it is super account then it will proceed to super user menu.

Super User Menu Interface

Register Admin Account

```
====================Welcome to Super User Menu====================

 1. Create an admin account
 2. Change password
 3. Log Out

 Enter your option 1/2/3 :
```

Figure 4.2.1.1

In figure 4.2.1 It shows super user menu that only reached when ID and password are both correct. Super User can do 2 command, Create a new admin account and change password.

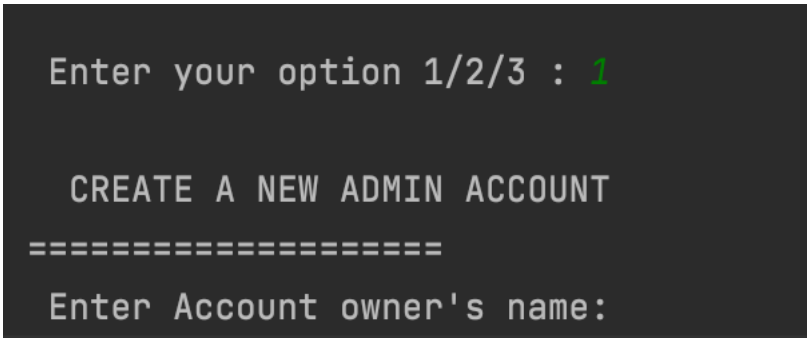


Figure 4.2.1.2

Then after choosing the option “1” it wil proceed and create an admin account.

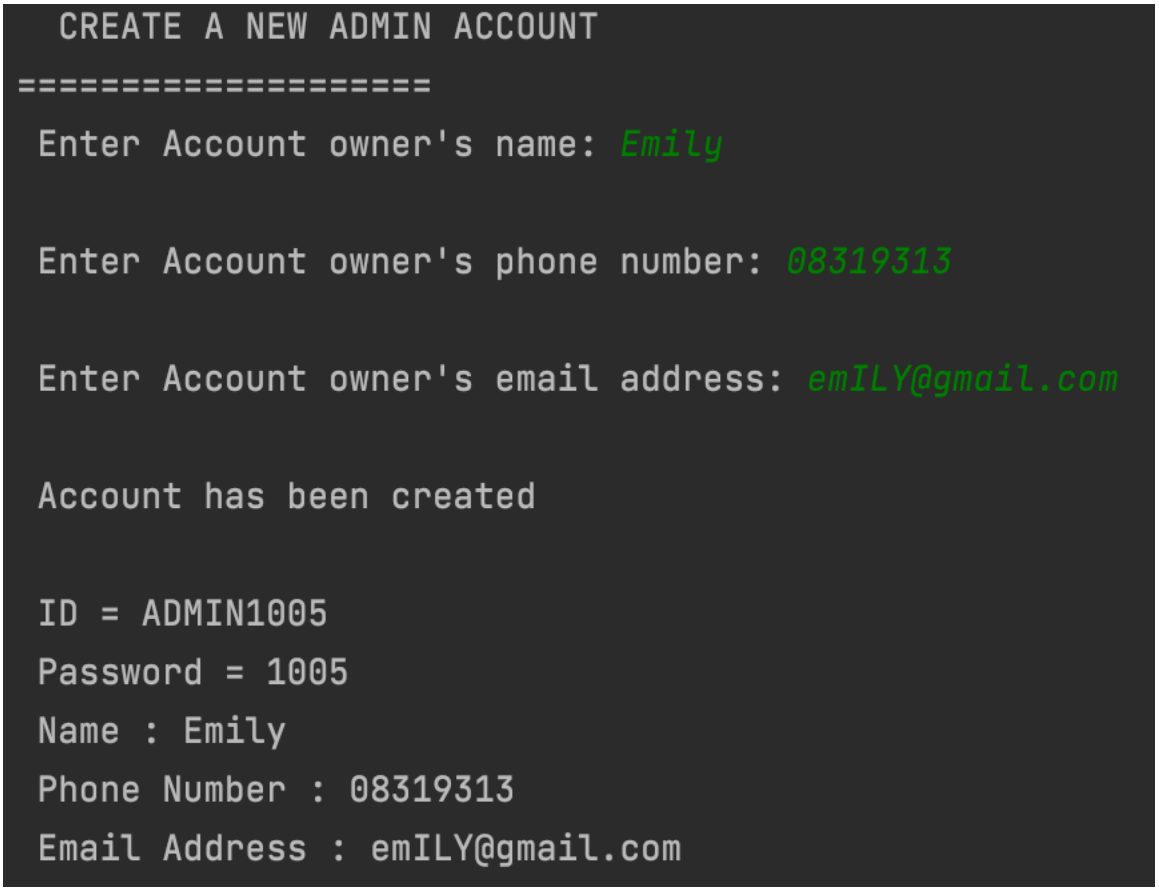


Figure 4.2.1.3

The system will ask for 3 input. Name, phone number, and email. Then the ID and password will be auto generated. Then, the output will be like this. Right after creating the admin account, the page will back to Super user Menu.

```
ID = ADMIN1005
Password = 1005
Name : Emily
Phone Number : 08319313
Email Address : emILY@gmail.com

====================Welcome to Super User Menu====================

1. Create an admin account
2. Change password
3. Log Out

Enter your option 1/2/3 :
```

Figure 4.2.1.4

Right after creating the admin account, the page will back to Super user Menu.

Change Password

```
Enter your option 1/2/3 : 2
Enter your current password: 1000
Enter a new password: 515604

Your password has been changed
```

Figure 4.2.2.1

The picture shown above is the interface when we choose option 2. When we entered wrong password then it is just will be like Figure 4.1.5. and when we entered the correct one we will have right to enter our new password.

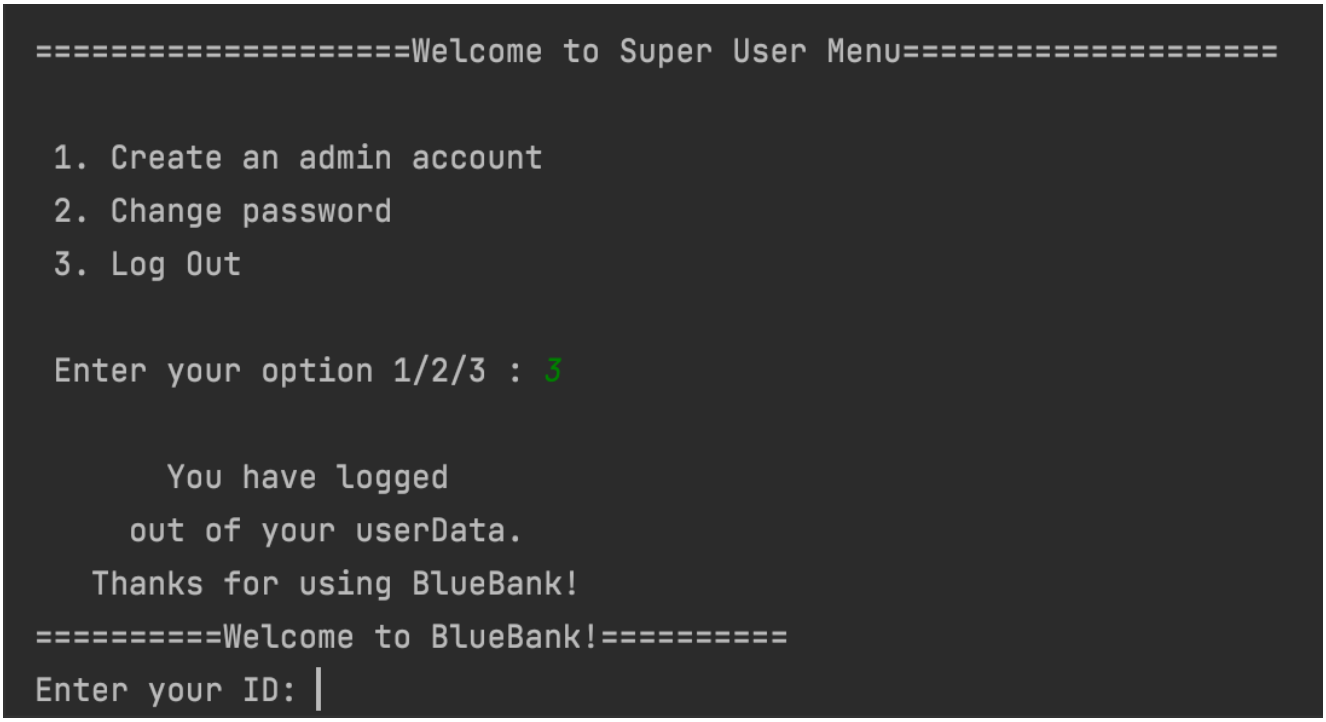


Figure 4.2.2.1

In order to finish using the banking system, don't forget to log out from you account by choosing "3" and the output will be like this.

Admin Interface

Register customer account

```
====================Welcome to Admin Staff Menu====================

 1. Create a new customer account
 2. Edit customer details
 3. See customer's statement of account report
 4. Change password
 5. Log Out

 Select an activity 1/2/3/4/5 : 1
```

Figure 4.3.1.1

To register customer account, all we have to do is just a single touch, input “1”

```
 Select an activity 1/2/3/4/5 : 1

 Which customer account are you planning to create?
 Savings account (CS) or Current account(CC): CS

  CREATE A NEW CUSTOMER ACCOUNT
====================
 Enter Account owner's name:
```

Figure 4.3.1.2

As in Figure 4.3.1.2 shown, first the system will ask whether the customer account we are going to create is saving account or current account. Then we are going to fill the registration form.

```
  CREATE A NEW CUSTOMER ACCOUNT
====================
 Enter Account owner's name: Mikae

 Enter Account owner's phone number: 02323232323

 Enter Account owner's email address: M1K43L@gmail.com

 Account has been created

 ID = SAVINGS1006
 Password = 1006
 Name : Mikae
 Phone Number : 02323232323
 Email Address : M1K43L@gmail.com
```

Figure 4.3.1.3

After filling the registration form, then the system will show us input of what we have inputted. Then we can use that ID and password to login.

Edit customer details

```
====================Welcome to Admin Staff Menu====================

 1. Create a new customer account
 2. Edit customer details
 3. See customer's statement of account report
 4. Change password
 5. Log Out

 Select an activity 1/2/3/4/5 : 2
```

Figure 4.3.2.1

To modify customer details, we have to go through option “2”

```
Select an activity 1/2/3/4/5 : 2

Enter the account number you wish to modify SAVINGS1006

1. ID :SAVINGS1006
2. Name : Mikae
3. Phone Number : 02323232323
4. Email Address : M1K43L@gmail.com
```

Figure 4.3.2.2

Then we will be shown our current information

```
Select an information you would like to change (3/4): 3
Enter customer's new phone number: 0817531422

Current phone number: 0817531422
```

Figure 4.3.2.3

Here in figure 4.3.2.3 we have to choose which data we want to change (3 for phone number, and 4 for email)

View customer's statement of account report

```
====================Welcome to Admin Staff Menu====================

 1. Create a new customer account
 2. Edit customer details
 3. See customer's statement of account report
 4. Change password
 5. Log Out

 Select an activity 1/2/3/4/5 : 3
```

Figure 4.3.3.1

```
====================Welcome to Admin Staff Menu====================

 1. Create a new customer account
 2. Edit customer details
 3. See customer's statement of account report
 4. Change password
 5. Log Out

 Select an activity 1/2/3/4/5 : 3
SAVINGS1002|Deposit RM500| on Monday, 13 December 2021 at 13:24:36|The current balance is now 800
SAVINGS1002|Withdrawal RM300| on Monday, 13 December 2021 at 13:27:06|The current balance is now 500
SAVINGS1002|Deposit RM100| on Monday, 13 December 2021 at 13:29:08|The current balance is now 600
start date:
```

Figure 4.3.3.2

It will start to print all statement of account report and asking for date.

```
start date: 13/12/2021 00:00:00
end date: 14/12/2021 23:59:59
SAVINGS1002|Deposit RM500| on Monday, 13 December 2021 at 13:24:36|The current balance is now 800

SAVINGS1002|Withdrawal RM300| on Monday, 13 December 2021 at 13:27:06|The current balance is now 500

SAVINGS1002|Deposit RM100| on Monday, 13 December 2021 at 13:29:08|The current balance is now 600
```

Figure 4.3.3.3

In a specific date the system will print the statement only in between those startdate and enddate.

Change Password

```
====================Welcome to Admin Staff Menu====================

 1. Create a new customer account
 2. Edit customer details
 3. See customer's statement of account report
 4. Change password
 5. Log Out

 Select an activity 1/2/3/4/5 : 4
 Enter your current password: 4040
 Enter a new password: 1001

 Your password has been changed

      Please login using
      your new password
==========Welcome to BlueBank!==========
Enter your ID:
```

Figure 4.3.4.1

In order to change password you have to input option 4, Then the system will ask you for your current password and input your new password. Then the system will restart and you can login with your new password

Customer Menu Interface

Perform deposit

```
==========Welcome to BlueBank!==========
Enter your ID: SAVINGS1006
Enter your password: 1006
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:1
```

Figure 4.4.1.1

To perform deposit we have to input option 1, if we input not from the option (not 1-7) the system will show an error like in figure 4.4.2

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:9

 The response you entered is invalid ...
```

Figure 4.4.1.2

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:1
 How much money you would like to deposit 300

 You have successfully deposit your balance!
```

Figure 4.4.1.3

The system will ask us to input how much money we want to deposit.

Perform withdrawal

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:2
 How much money you would like to withdraw 500

 You have successfully withdraw your balance!
```

Figure 4.4.2.1

Perform transfer

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:3
Enter the account number you would like to transfer .

The ID you entered is wrong

Enter the account number you would like to transfer
```

Figure 4.4.3.1

When we choose option 3 we are going to proceed to the next statement which ask us to input our destination. If we input it wrong the output is “The ID you entered is wrong” and we have to input the correct one.

```
Enter the account number you would like to transfer SAVINGS1002
 How much money you would like to transfer 300

 You have successfully transfer your balance!
```

Figure 4.4.3.2

Then the account number we use to transfer will minus its balance, and the destination will be added a number of the transfer amount (in Figure 4.4.3.2) are 300.

Check Balance

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:4

Type 1 to check user balance
     2 to back
1
Your current balance is :500
```

Figure 4.4.4.1

User can check their balance by pressing 4,

Then press 1 to check their current balance after all the transaction they made

View Account report

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:5
SAVINGS1006|Deposit RM300| on Monday, 13 December 2021 at 13:57:57|The current balance is now 400

SAVINGS1006|Deposit RM300| on Monday, 13 December 2021 at 14:02:14|The current balance is now 700

SAVINGS1006|Deposit RM300| on Monday, 13 December 2021 at 14:03:01|The current balance is now 1000

SAVINGS1006|Withdrawal RM500| on Monday, 13 December 2021 at 14:03:57|The current balance is now 500

SAVINGS1006|Transfer RM300| on Monday, 13 December 2021 at 14:12:55|The current balance is now 200

start date:
```

Figure 4.4.5.1

When we input 5, all of the customer transaction history will be printed. Then we have the option to find it in a specific date in order to make us easier when looking for our transaction in some particular events.

```
start date: 13/12/2021 00:00:00
end date: 13/12/2021 14:03:00
SAVINGS1006|Deposit RM300| on Monday, 13 December 2021 at 13:57:57|The current balance is now 400

SAVINGS1006|Deposit RM300| on Monday, 13 December 2021 at 14:02:14|The current balance is now 700
```

Figure 4.4.5.2

This is the example, I only take from 13 December 2021 00:00:00 until 13 December 2021 14:03:00 then it showed only 2 transactions history

Change password

```
====================Welcome to Customer Menu====================

 1. Deposit
 2. Withdrawal
 3. Transfer
 4. Check Balance
 5. View Account Report
 6. Change Password
 7. Log Out
 Enter your option 1/2/3/4/5/6/7:6
 Enter your current password: 1006
 Enter a new password: 6001

 Your password has been changed

      Please login using
      your new password
```

Figure 4.4.6.1

User must enter their old password first in order to change their password, then input a new password and login with the new password.

Conclusion

In brief, the world has evolved rapidly over the years. By learning logical things like python programming especially in the making of banking system is not as simple as that. All of this are one unity and a very complex program for me to understood honestly. But with the help of flowchart and pseudocode, the complexity level is decreased and easier to be understood. Since then flowchart and pseudocode is look like a small things but important enough. Program source code contains many syntax that comes from evolving the pseudocode and the flowchart. Last but not least I feel really grateful for finishing this task in time.

References

B. A., C. S. (n.d.). *What Is Source Code?* ThoughtCo. https://www.thoughtco.com/source-code-definition-958200#:~:text=Source%20code%20is%20the%20list

Python Variables. (n.d.). Www.w3schools.com. https://www.w3schools.com/python/python_variables.asp

What is an Operator? - Definition from Techopedia. (n.d.). Techopedia.com. https://www.techopedia.com/definition/3485/operator-programming

Definition of MULTIPLICATION. (n.d.). Www.merriam-Webster.com. https://www.merriam-webster.com/dictionary/multiplication

Python Functions. (2018, September 8). GeeksforGeeks. https://www.geeksforgeeks.org/python-functions/

Python Functions. (2019). W3schools.com. https://www.w3schools.com/python/python_functions.asp

Python While Loops. (n.d.). Www.w3schools.com. https://www.w3schools.com/python/python_while_loops.asp