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	**:**	**LIM ZI HONG**
TP NUMBER	**:**	**TP064970**

<u>INSTRUCTIONS TO CANDIDATES:</u>

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

1.0 Introduction

Technology has exponentially improved after World War II, particularly in communication technology through the rapid construction of radio towers, linking every city, from all over the world. Up until this point in time, communication has never been easier, with a few clicks of a button, we're able to talk to anyone, be it from anywhere in the world that has an active connection to satellites. An improvement in communication technology has only contributed to improving humanity's quality of life, and ultimately, result in further development of technology in all fields. As such, this has directly led to an increase in global trading, mandating the necessity of banking systems in which people are able to pay large amounts of cash without the need to bring a single penny on them. Furthermore, this also enables currency to be exchanged at will, facilitating international trading in which huge sums of money can be transferred and switched to a different currency at the same time.

1.1 Assumption

1) I assume that the system for customer account only can withdraw and deposit.

2) I assume that only name, IC number, email and phone number are required to open account.

3) I assume that username of bank account is number that arrange sequence.

4) I assume that old password is not required to input to change password.

5) I assume that 0 can be input in withdraw and deposit.

6)I assume that bank statement is printed this month of statement and last month of statement.

7)I assume that balance not enough to open bank account will back to account option.

2.0 Design of Program - Pseudocode

Auto generate customer id

```
DEFINE gen_new_cusid ()
    DECLARE (STRING: lines, line, default_no, no, nextid), (INTEGER: i)
     i=1
    SET default_no = "00000"
    WITH OPEN “user.txt” with READ mode as cus_no:
        lines = cus_no.readlines
    ENDWITH
    LOOP line IN lines:
        IF line [:3] = "CUS" THEN
            i = i+1
        ENDIF
    ENDLOOP
    no = CONVERT i to STRING
    nextid = "CUS" + default_no [: LENGTH of default_no -LENGTH of no] + no
    RETURN nextid
ENDDEFINE
```

Auto generate staff id

```
DEFINE gen_new_staffid ()
    DECLARE (STRING: lines, line, default_no, no, nextid), (INTEGER: i)
     i=1
    SET default_no = "00000"
    WITH OPEN “user.txt” with READ mode as staff_no:
        lines = staff_no.readlines ()
    ENDWITH
    LOOP line IN lines:
        IF line [:5] = "STAFF" THEN
            i = i+1
        ENDIF
    ENDLOOP
    no = CONVERT i to STRING
    nextid = "STAFF" + default_no [: LENGTH of default_no -LENGTH of no] + no
    RETURN nextid
ENDDEFINE
```

Staff and customer modify password

```
DEFINE change password ():
    DECLARE (STRING: userdetails, userdetail, username, recs, frec, field, newpassword),
        (LIST: edit_list, rec)
    WITH OPEN "user.txt" with Read mode as read
        userdetails = read. readlines ()
    END WITH
    LOOP userdetail IN userdetails:
        rec = userdetail. split (":")
        IF username =rec [0] THEN
            DISPLAY ("NEW PASSWORD:")
            READ newpassword
            ASSIGN rec [1] = newpassword
        ENDIF
        APPEND edit_list with rec
    ENDLOOP
    WITH OPEN "user.txt" with WRITE mode as edit:
        LOOP recs IN edit_list:
            DECLARE frec = ""
            LOOP field IN recs:
                frec += field + ":"
            WRITE ":" with frec.strip
    ENDWITH
        DISPLAY "Successfully change password"
ENDEFINE
```

Customer view details

```
DEFINE customer_detail ():
    DECLARE (STRING: username, customerdetail, customerdetails), (LIST: rec)
    WITH OPEN "user.txt" with READ mode as customeread
        customerdetails = customeread.readlines()
    LOOP customerdetail IN customerdetails:
        rec = customerdetail.split(":")
        IF username = rec [0] THEN
            DISPLAY ("====================")
            DISPLAY ("ACCOUNT DETAILS")
            DISPLAY ("====================")
            DISPLAY ("ACCOUNT TYPE        :" + rec [2])
            DISPLAY ("ACCOUNT BALANCE :" + rec [3])
            DISPLAY ("ACCOUNT NO          :" + rec [0])
            DISPLAY ("NAME                :" + rec [4])
            DISPLAY ("IDENTITY CARD       :" + rec [5])
            DISPLAY ("PHONE NO            :" + rec [6])
            DISPLAY ("EMAIL               :" + rec [7])
        ENDIF
    ENDLOOP

ENDEFINE
```

Customer deposit

```
DEFINE customer_deposit ():
    DECLARE (STRING: username, year, month, day, local_time, deposits, recs, frec, field),
            (LIST: edit_list, rec, newtrans), (INTEGER: now, deposit)
    SET edit_list = []
    FROM datetime IMPORT datetime
    now = datetime.datetime.now ()
    year = CONVERT now.year to STRING
    month = CONVERT now.month to STRING
    day = CONVERT now.day to STRING
    local_time = (year + "-" + month + "-" +day)
    DOWHILE TRUE:
        TRY:
            DISPLAY("DEPOSIT:")
            READ deposit
            deposits = CONVERT deposit to STRING
            BREAK
        ENDTRY
        EXCEPT ValueError:
            DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
```

```
                CONTINUE
        ENDEXCEPT
ENDDO

WITH OPEN "user.txt" with READ mode as customeread
        customerdetails = customeread.readlines()
ENDWITH

LOOP customerdetail IN customerdetails:
        rec = customerdetail.split(":")
        IF username = rec [0] THEN
                balance = CONVERT rec [3] TO INTEGER
                ASSIGN rec [3] = CONVERT (balance+ deposit) to STRING
                WITH OPEN "customerstatement.txt" WITH APPEND as writetrans
                        SET newtrans = [local_time,"DEPOSIT", username, deposits, rec [3]]
                        newtrans = ":".join(newtrans)
                        APPEND "customerstatement.txt" with (newtrans + "\n")
                ENDWITH
        ENDIF
        APPEND edit_list with rec
ENDLOOP
```

```
    WITH OPEN "user.txt" with WRITE mode as edit:
        LOOP recs IN edit_list:
            DECLARE frec = ""
            LOOP field IN recs:
                frec += field + ":"
            WRITE ":" with frec.strip
    ENDWITH

    DISPLAY "Successfully Deposit"
ENDDEFINE
```

Customer withdrawal

```
DEFINE customer_withdrawal ():
    DECLARE (STRING: username, year, month, day, local_time, withdrawals, recs, frec, field),
        (LIST: edit_list, rec, newtrans), (INTEGER: now, withdrawal, flag)
    FROM datetime IMPORT datetime
    SET edit_list = []
    flag = 1
    now = datetime.datetime.now ()
    year = CONVERT now.year to STRING
    month = CONVERT now.month to STRING
    day = CONVERT now.day to STRING
    local_time = (year + "-" + month + "-" +day)
    DOWHILE TRUE:
        TRY:
            DISPLAY("WITHDRAWAL:")
            READ withdrawal
            withdrawals = CONVERT withdrawal to STRING
            BREAK
        ENDTRY
        EXCEPT ValueError:
            DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
            CONTINUE
        ENDEXCEPT
    ENDDO

    WITH OPEN "user.txt" with READ mode as customeread
        customerdetails = customeread.readlines()
    ENDWITH

    LOOP customerdetail IN customerdetails:
        rec = customerdetail.split(":")
        IF username = rec [0] THEN
            balance = CONVERT rec [3] TO INTEGER
            IF rec [2] = "SAVING ACCOUNT"THEN
                IF (balance - withdrawal) <100 THEN
                    DISPLAY ("THIS WITHDRAWAL AMOUNT HAS AFFECT MINIMUM BALANCE")
                    flag = 0
                ELSE:
                    rec [3] = CONVERT (balance - withdrawal) to STRING
                    WITH OPEN "customerstatement.txt" WITH APPEND as writetrans
                        newtrans = [local_time,"DEPOSIT”, username, deposits, rec [3]]
                        newtrans = ":”. join(newtrans)
                        APPEND "customerstatement.txt" with (newtrans + "\n")
                    ENDWITH
                ENDIF

            ELIF rec [2] = "CURRENT ACCOUNT" THEN
                IF (balance - withdrawal) <500 THEN
                    DISPLAY ("THIS WITHDRAWAL AMOUNT HAS AFFECT MINIMUM BALANCE")
                    flag = 0
                ELSE:
                    rec [3] = CONVERT (balance - withdrawal) to STRING
                    WITH OPEN "customerstatement.txt" WITH APPEND as writetrans
                        newtrans = [local_time,"DEPOSIT”, username, deposits, rec [3]]
                        newtrans = ":”. join(newtrans)
                        APPEND "customerstatement.txt" with (newtrans + "\n")
                    ENDWITH
                ENDIF
            ENDIF
        ENDIF
        APPEND edit_list with rec
```
```
ENDLOOP
```

```
    IF flag =1 THEN
        WITH OPEN "user.txt" with WRITE mode as edit:
            LOOP recs IN edit_list:
                DECLARE frec = ""
                LOOP field IN recs:
                    frec += field + ":"
                WRITE ":" with frec.strip
        ENDWITH
        DISPLAY "Successfully Withdrawal"
    ENDIF
ENDDEFINE
```

Customer print bank statement

```
DEFINE print_bank_state():
    DECLARE (STRING: year, years, month, months, day, details, detail, username, trans, tran, total_dep, total_with),
        (INTEGERS: totalwith, totaldep, balance), (LIST: rec, rect)
    FROM datetime IMPORT datetime
    now = datetime.datetime.now()
    totalwith = 0
    totaldep = 0
    year =  CONVERT now.year to STRING
    years = CONVERT (now.year - 1) to STRING
    month = CONVERT now.month to STRING
    months = CONVERT (now.month - 1) to STRING
    IF LENGTH of months=1 THEN
        SET months = "0" + months
    ENDIF

    IF month = "03" OR month = "05" OR month = "07" OR month = "08" OR month = "10" OR month = "12 THEN
        SET day ="31"

    ELIF month = "02" or month = "04" or month = "06" or month = "09" or month = "11" THEN
```

```
        SET day ="30"
    ENDIF
    DISPLAY ("==============================")
    DISPLAY ("BANK STATMENT")
    DISPLAY ("==============================")

    WITH OPEN "user.txt" with READ mode as detailread:
        details = detailread.readlines()
    ENDWITH
    LOOP detail IN details:
        rec = detail. split(":")
        IF username = rec [0] THEN
            DISPLAY (rec [4])
            DISPLAY ("ACCOUNT NO:" + rec [0])
        ENDIF
    ENDLOOP

    IF month = "01" THEN
        DISPLAY ("STATEMENT PERIOD:" + years + "/" + "12" + "/" + "01" + "-" + year + "/" + month + "/" + "31")
    ELSE:
        DISPLAY ("STATEMENT PERIOD:" + year + "/" + months + "/" + "01" + "-" + years + "/" + month + "/" + day)
```

```
ENDIF


DISPLAY ("--------------------------------------------------------------")

DISPLAY ("DATE".center(20),"DEPOSIT".center(20),"WITHDRAWAL".center(20),"BALANCE".center(20))

DISPLAY ("--------------------------------------------------------------")

WITH OPEN "customerstatement.txt" with READ mode as transread:

    trans = transread.readlines()

ENDWITH

LOOP tran IN trans:

    rect = tran.split(":")

    IF rect[2] = username THEN

        IF month = "01" THEN

            IF (rect[0])[:7] =(year + "-" + month) or (rect[0])[:7] = (years + "-" + "12") THEN

                balance = CONVERT rect[3] to INTEGER

                IF rect[1] = "DEPOSIT"THEN

                    DISPLAY (rect[0].center(20),rect[3].center(20),"".center(20),rect[4].center(20))

                    totaldep = totaldep + balance

                ELIF rect[1] = "WITHDRAWAL" THEN

                    DISPLAY(rect[0].center(20),"".center(20),rect[3].center(20),rect[4].center(20))
                    totalwith = totalwith + balance

                ENDIF
```

```
                    ENDIF
                ELSE:
                    IF(rect[0])[:7] =(year+ "-" + month) or (rect[0])[:7] = (year + "-" + months) THEN
                        balance = CONVERT rect[3] to INTEGER
                        IF rect[1] = "DEPOSIT" THEN
                            DISPLAY (rect[0].center(20),rect[3].center(20),"".center(20),rect[4].center(20))
                            totaldep = totaldep + balance

                        ELIF rect[1] = "WITHDRAWAL"THEN
                            DISPLAY(rect[0].center(20),"".center(20),rect[3].center(20),rect[4].center(20))
                            totalwith = totalwith + balance
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
    ENDLOOP
    total_dep = CONVERT totaldep to STRING
    total_with = CONVERT totalwith to STRING
    DISPLAY ("TOTAL WITHDRAWAL:" + total_with)
    DISPLAY ("TOTAL DEPOSIT   :" + total_dep)
ENDDEFINE
```

Staff create customer account

```
DEFINE create_customer_acc():
    DECLARE (STRING:user_account_key, user_account, user_name, user_IC, user_phoneno, user_email, user_password, next_id),
            (INTEGERS: user_balance), (LIST: new_cus)
    DISPLAY ("====================")
    DISPLAY ("CREATE CUSTOMER ACCOUNT")
    DISPLAY ("====================")
    DOWHILE TRUE:
        DISPLAY ("1) CURRENT ACCOUNT 2) SAVING ACOUNT ,3) EXIT PLEASE SELECT (1,2,3):")
        READ user_account_key
        IF user_account_key = "1" THEN
            SET user_account = "CURRENT ACCOUNT"
            DOWHILE TRUE:
                TRY:
                    DISPLAY ("MIN BALANCE IS RM500, PLEASE ENTER AMOUNT:"))
                    READ user_balance
                    BREAK
                ENDTRY

                EXCEPT ValueError:
```

```
        DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
        CONTINUE
    ENDEXCEPT
ENDDO

IF user_balance < 500 THEN
    DISPLAY ("INSUFFICIENT AMOUNT")
    CONTINUE
ELIF user_balance >= 500 THEN
    DISPLAY("CUSTOMER NAME:")
    READ user_name
    DISPLAY ("IDENTITY NUMBER:")
    READ user_IC
    DISPLAY ("PHONE NUMBER:")
    READ user_phoneno
    DISPLAY ("EMAIL:")
    READ user_email
    user_password = (user_name + "@" + user_IC[:5] + user_email[:5])
    WITH OPEN "user.txt" with APPEND mode as cusdetail:
        nextid = gen_new_cusid()
        newcus = [nextid,user_password,user_account,str(user_balance),
                    user_name,user_IC,user_phoneno,user_email]
                newcus = ':'.join(newcus)
                APPEND "user.txt" with (newcus + "\n")
            DISPLAY ("Customer Username:" + nextid)
            DISPLAY ("Customer Password:" + user_password)
            DISPLAY ("CUSTOMER ACCOUNT CREATED")
            BREAK
        ENDIF
    ELIF user_account_key = "2" THEN
        user_account = "SAVING ACCOUNT"
        DOWHILE TRUE:
            TRY:
                DISPLAY ("MIN BALANCE IS RM100, PLEASE ENTER AMOUNT:"))
                READ user_balance
                BREAK
            ENDTRY

            EXCEPT ValueError:
                DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
                CONTINUE
            ENDEXCEPT
```

```
IF user_balance < 100 THEN
    DISPLAY ("INSUFFICIENT AMOUNT")
    CONTINUE
ELIF user_balance >= 100 THEN
    DISPLAY("CUSTOMER NAME:")
    READ user_name
    DISPLAY ("IDENTITY NUMBER:")
    READ user_IC
    DISPLAY ("PHONE NUMBER:")
    READ user_phoneno
    DISPLAY ("EMAIL:")
    READ user_email
    user_password = (user_name + "@" + user_IC[:5] + user_email[:5])
    WITH OPEN "user.txt" with APPEND mode as cusdetail:
        nextid = gen_new_cusid()
        newcus = [nextid,user_password,user_account,str(user_balance),
                  user_name,user_IC,user_phoneno,user_email]
        newcus = ':'.join(newcus)
        APPEND "user.txt" with (newcus + "\n")
    DISPLAY ("Customer Username:" + nextid)
```

```
                DISPLAY ("Customer Password:" + user_password)
                DISPLAY ("CUSTOMER ACCOUNT CREATED")
                BREAK
            ENDIF
        ELIF user_account_key = "3" THEN
            BREAK
        ELSE:
            print ("INVALID ACTION")
            CONTINUE
        ENDIF
    ENDDO
ENDDEFINE
```

Staff edit customer details

```
DEFINE edit_customer_detail():
    DECLARE (STRING: user_name, cusdetails, cusdetail, action, phoneno, email, recs, frec, field),
            (INTEGERS: flag), (LIST:edit_list, rec)
    flag = 0
    DISPLAY ("Customer Account No:")
    READ user_name
    WITH OPEN "user.txt" with READ mode as read:
        cusdetails = read.readlines()
    ENDWITH
    LOOP cusdetail IN cusdetails:
        rec = cusdetail.split(":")
        IF user_name = rec [0] and user_name [:3] = "CUS" THEN
            DISPLAY ("1) PHONE NO:" + rec [6])
            DISPLAY ("2) EMAIL      :" + rec[7])
            SET flag =1
            DOWHILE TRUE:
                DISPLAY ("PLEASE SELECT (1) PHONE NO, (2) EMAIL TO EDIT:")
                READ action
                IF action = "1" THEN
```

```
                    DISPLAY ("NEW PHONE NO:")
                    READ phoneno
                    ASSIGN rec[6] = phoneno
                    DISPLAY ("Successfully updated!")
                    BREAK
                ELIF action = "2" THEN
                    DISPLAY ("NEW EMAIL:")
                    READ email
                    ASSIGN rec [7] = email
                    DISPLAY ("Successfully updated!")
                    BREAK
                ELSE:
                    DISPLAY ("INVALID ACTION")
                    CONTINUE
                ENDIF
            ENDDO
        ENDIF
        APPEND edit_list with rec
    ENDLOOP
    IF flag =1 THEN
        WITH OPEN "user.txt" with WRITE mode as edit:
            LOOP recs IN edit_list:
                DECLARE frec = ""
                LOOP field IN recs:
                    frec += field + ":"
                WRITE ":" with frec.strip
        ENDWITH

    ELIF flag = 0 THEN
        DISPLAY ("THIS ACCOUNT IS NOT AVAILABLE")
    ENDIF
ENDDEFINE
```

Staff print customer bank statement

```
DEFINE staff_print_bank_state():
    DECLARE (STRING: account,year, years, month, months, day, details, detail, username, trans, tran, total_dep, total_with),
            (INTEGERS: flag, totalwith, totaldep, balance), (LIST: rec, rect)
    FROM datetime IMPORT datetime
    DISPLAY ("ACCOUNT NO")
    READ account
    now = datetime.datetime.now()
    flag = 0
    totalwith = 0
    totaldep = 0
    year =  CONVERT now.year to STRING
    years = CONVERT (now.year - 1) to STRING
    month = CONVERT now.month to STRING
    months = CONVERT (now.month - 1) to STRING
    IF LENGTH of months=1:
        SET months = "0" + months
    ENDIF
```

```
IF month = "03" OR month = "05" OR month = "07" OR month = "08" OR month = "10" OR month = "12 THEN
    SET day ="31"

ELIF month = "02" or month = "04" or month = "06" or month = "09" or month = "11" THEN
    SET day ="30"
ENDIF

WITH OPEN "user.txt" with READ mode as detailread:
    details = detailread.readlines()
ENDWITH
LOOP detail IN details:
    rec = detail. split(":")
    IF account = rec [0] THEN
        DISPLAY ("==============================")
        DISPLAY ("BANK STATMENT")
        DISPLAY ("==============================")
        DISPLAY (rec [4])
        DISPLAY ("ACCOUNT NO:" + rec [0])
        IF month = "01" THEN
            DISPLAY ("STATEMENT PERIOD:" + years + "/" + "12" + "/" + "01" + "-" + year + "/" + month + "/" + "31")
```

```
                ELSE:
                        DISPLAY ("STATEMENT PERIOD:"+ year + "/" + months + "/" + "01" + "-" + years + "/" + month + "/" + day)
                ENDIF
                DISPLAY ("------------------------------------------------------------")
                DISPLAY ("DATE".center(20),"DEPOSIT".center(20),"WITHDRAWAL".center(20),"BALANCE".center(20))
                DISPLAY ("------------------------------------------------------------")
                BREAK
        ENDIF
ENDLOOP
IF account! =rec [0] THEN
        DISPLAY ("THIS ACCOUNT IS NOT AVAILABLE")
        SET flag = 1
ENDIF

WITH OPEN "customerstatement.txt" with READ mode as transread:
        trans = transread.readlines()
ENDWITH

LOOP tran IN trans:
        rect = tran.split(":")
```

```
IF rect[2] = account THEN

    IF month = "01" THEN

        IF (rect[0])[:7] ==(year + "-" + month) or (rect[0])[:7] == (years + "-" + "12") THEN

            balance = CONVERT rect[3] to INTEGER

            IF rect[1] == "DEPOSIT"THEN

                DISPLAY (rect[0].center(20),rect[3].center(20),"".center(20),rect[4].center(20))

                totaldep = totaldep + balance

            ELIF rect[1] == "WITHDRAWAL" THEN

                DISPLAY(rect[0].center(20),"".center(20),rect[3].center(20),rect[4].center(20))
                totalwith = totalwith + balance

            ENDIF

        ENDIF


    ELSE:

        IF(rect[0])[:7] ==(year+ "-" + month) or (rect[0])[:7] == (year + "-" + months) THEN

            IF rect[1] = "DEPOSIT" THEN

                DISPLAY (rect[0].center(20),rect[3].center(20),"".center(20),rect[4].center(20))

                totaldep = totaldep + balance


            ELIF rect[1] = "WITHDRAWAL"THEN
```

```
                        DISPLAY(rect[0].center(20),"".center(20),rect[3].center(20),rect[4].center(20))
                        totalwith = totalwith + balance
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDLOOP

    IF flag = 0 THEN
        total_dep = CONVERT totaldep to STRING
        total_with = CONVERT totalwith to STRING
        DISPLAY ("TOTAL WITHDRAWAL:" + total_with)
        DISPLAY ("TOTAL DEPOSIT   :" + total_dep)
    ENDIF
ENDDEFINE
```

Super user creates staff account

```
DEFINE create_staff_acc():
    DECLARE (STRING: user_name, user_IC, user_phoneno, user_email, user_password, nextid),
        (LIST: newstaff)
    DISPLAY ("=====================")
    DISPLAY ("CREATE STAFF ACCOUNT")
    DISPLAY ("=====================")
    DISPLAY ("STAFF NAME:")
    READ user_name
    DISPLAY ("IDENTITY NUMBER:")
    READ user_IC
    DISPLAY ("PHONE NUMBER:")
    READ user_phoneno
    DISPLAY ("EMAIL:")
    READ user_email
    user_password = (user_name + "@" + user_IC[:5] + user_email[:5])
    WITH OPEN "user.txt" with APPEND as staffdetail:
        nextid = gen_new_staffid()
        SET newstaff = [nextid,user_password,user_name,user_phoneno,user_IC,user_email]
        newstaff = ":”.join(newstaff)
        APPEND "user.txt" with (newstaff + "\n")
    ENDWITH
    DISPLAY ("Staff Username:" + nextid)
    DISPLAY ("Staff Password:" + user_password)
    DISPLAY ("STAFF ACCOUNT CREATED")
ENDDEFINE
```

Customer account menu

```
DEFINE customer_acc_menu():
    DECLARE (STRING: action)
    DISPLAY ("====================")
    DISPLAY ("WELCOME")
    DOWHILE TRUE:
        DISPLAY ("====================")
        DISPLAY ("1.DETAIL")
        DISPLAY ("2.PRINT BANK STATEMENT")
        DISPLAY ("3.DEPOSIT")
        DISPLAY ("4.WITHDRAWAL")
        DISPLAY ("5.CHANGE PASSWORD")
        DISPLAY ("6.QUIT")
        DISPLAY ("PLEASE SELECT (1,2,3,4,5,6):")
        READ action
        IF action = "1" THEN
            customer_detail()
        ELIF action = "2" THEN
            print_bank_state()
        ELIF action = "3" THEN
            customer_deposit()
        ELIF action = "4" THEN
            customer_withdrawal()
        ELIF action = "5" THEN
            change_password()
        ELIF action = "6" THEN
            DISPLAY ("Goodbye")
            DISPLAY ("====================")
            BREAK
        ELSE:
            DISPLAY ("invalid action")
            CONTINUE
        ENDIF
    ENDDO
ENDDEFINE
```

Staff account menu

```
DEFINE staff_acc_menu():
    DECLARE (STRING: action)
    DISPLAY ("=====================")
    DISPLAY ("WELCOME")
    DOWHILE TRUE:
        DISPLAY ("=====================")
        DISPLAY ("1.CREATE CUSTOMER ACCOUNT")
        DISPLAY ("2.PRINT CUSTOMER BANK STATEMENT")
        DISPLAY ("3.EDIT CUSTOMER DETAILS")
        DISPLAY ("4.CHANGE PASSWORD")
        DISPLAY ("5.QUIT")
        DISPLAY ("PLEASE SELECT (1,2,3,4,5):")
        READ action
        IF action = "1" THEN
            create_customer_acc()
        ELIF action = "2" THEN
            staff_print_bank_state()
        ELIF action = "3" THEN
            edit_customer_detail()
        ELIF action = "4" THEN
            change_password()
        ELIF action = "5"THEN
            DISPLAY ("Goodbye")
            DISPLAY ("=====================")
            BREAK
        ELSE:
            DISPLAY ("invalid action")
            CONTINUE
        ENDIF
    ENDDO
ENDDEFINE
```

Super user account menu

```
DEFINE super_acc_menu():
    DECLARE (STRING: action)
    DISPLAY ("=====================")
    DISPLAY ("WELCOME")
    DOWHILE TRUE:
        DISPLAY ("=====================")
        DISPLAY ("1.CREATE STAFF ACCOUNT")
        DISPLAY ("2.QUIT")
        DISPLAY ("PLEASE SELECT (1,2):")
        READ action
        IF action = "1" THEN
            create_staff_acc()
        ELIF action = "2" THEN
            DISPLAY ("GOODBYE")
            DISPLAY ("=====================")
            BREAK
        ELSE:
            DISPLAY ("invalid action")
            CONTINUE
        ENDIF
    ENDDO
ENDDEFINE
```

User login

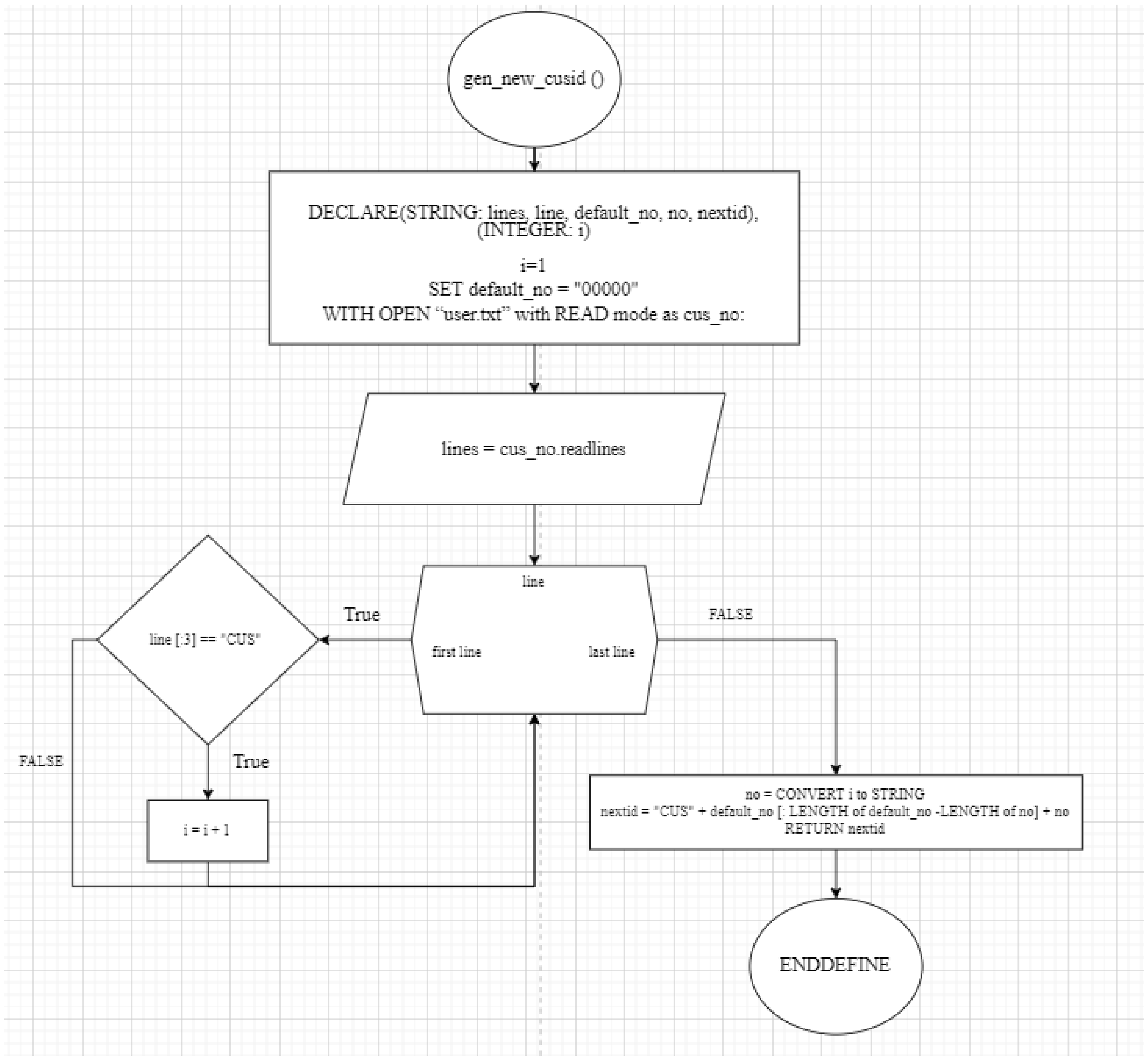
```
DEFINE login():
    DECLARE (STRING: user_reads, user_read, username, password), (LIST: usr)
    WITH OPEN "user.txt" with READ mode as login_read:
        user_reads = login_read.readlines()
    LOOP user_read IN user_reads:
        usr = user_read.split(":")
        IF username = usr[0] AND password = usr[1] THEN
            IF username[:5] = "SUPER" THEN
                super_acc_menu()
            ELIF username[:5] = "STAFF" THEN
                staff_acc_menu()
            ELIF username[:3] = "CUS" THEN
                customer_acc_menu()
            ENDIF
            BREAK
        ENDIF
    ENDLOOP
    IF username ! = usr[0] OR (password) != usr[1] THEN
        PRINT ("INVALID USERNAME OR PASSWORD")
    ENDIF
ENDDEFINE
```

Main logic

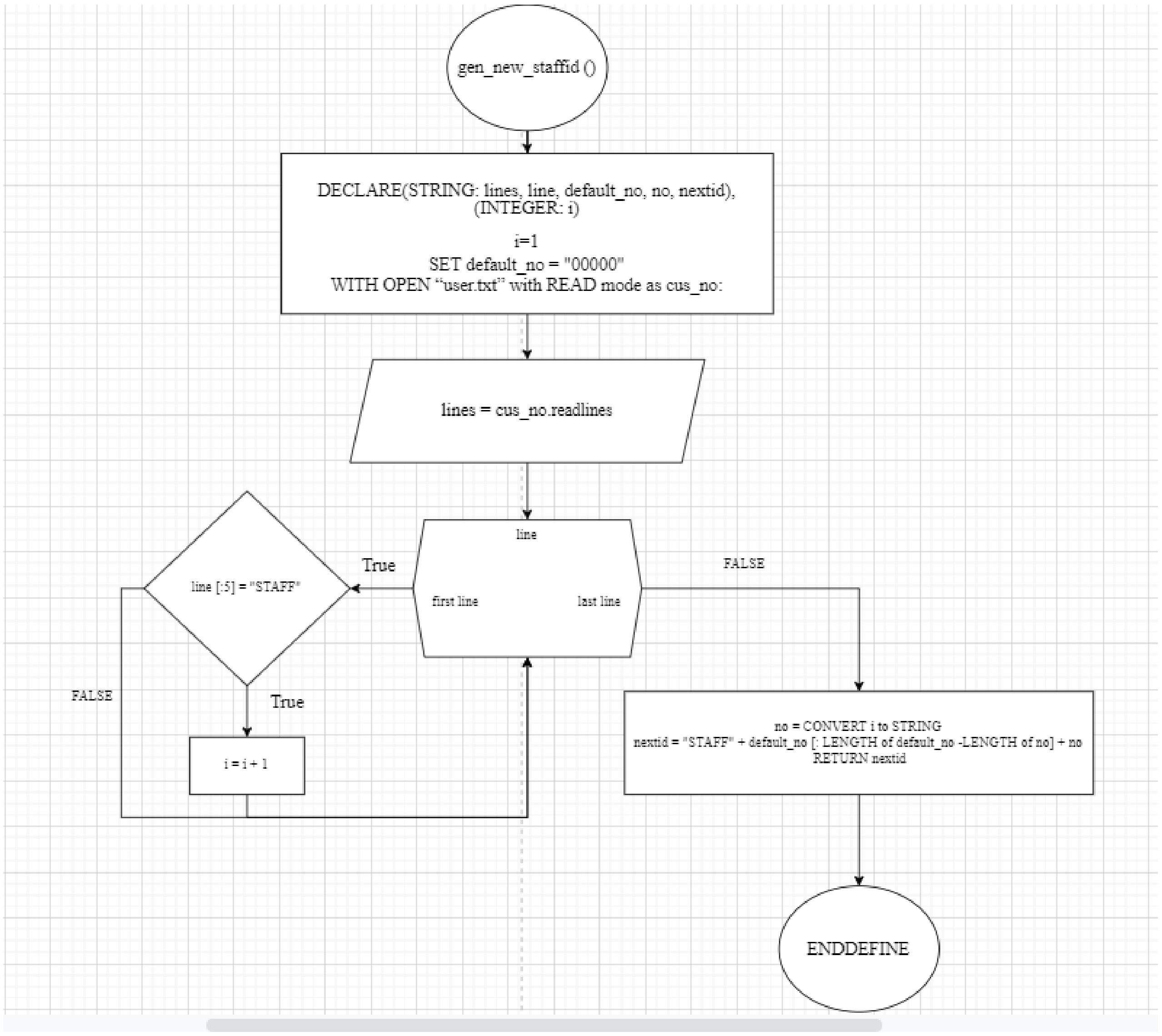
```
DECLARE (STRING: username, password)
DOWHILE TRUE
    DISPLAY ("=====================")
    DISPLAY ("*********************")
    DISPLAY ("BANKING SERVICE LOGIN")
    DISPLAY ("*********************")
    DISPLAY ("=====================")
    DISPLAY ("USERNAME:")
    READ username
    DISPLAY ("PASSWORD:")
    READ password
    login()
ENDDO
```

2.1 Design of Program - Flow Chart

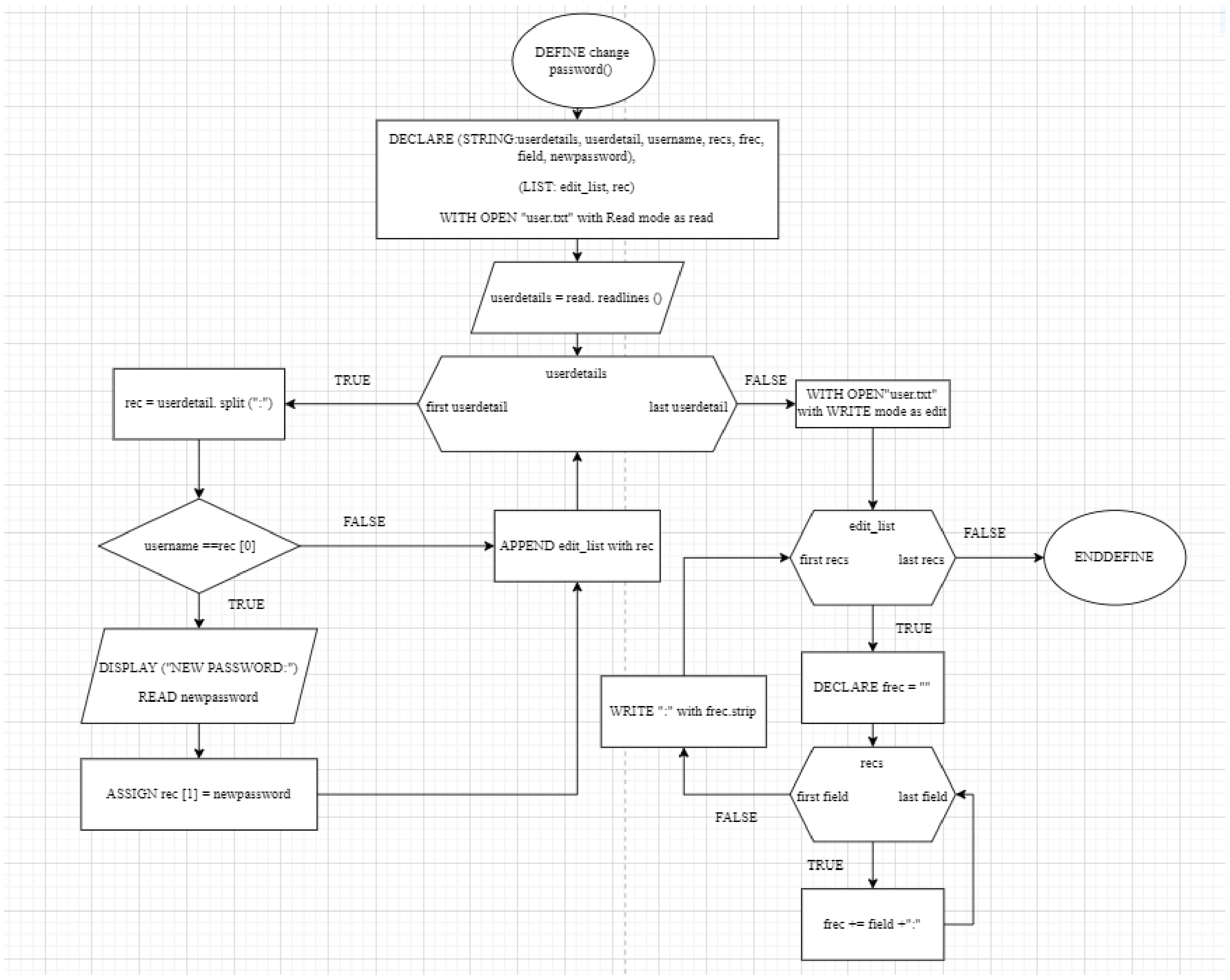
Auto generate customer id


gen_new_cusid ()
DECLARE(STRING: lines, line, default_no, no, nextid), (INTEGER: i)
i=1
SET default_no = "00000"
WITH OPEN "user.txt" with READ mode as cus_no:
lines = cus_no.readlines
line
first line
last line
True
FALSE
line [:3] == "CUS"
True
FALSE
i = i + 1
no = CONVERT i to STRING
nextid = "CUS" + default_no [: LENGTH of default_no -LENGTH of no] + no
RETURN nextid
ENDDEFINE

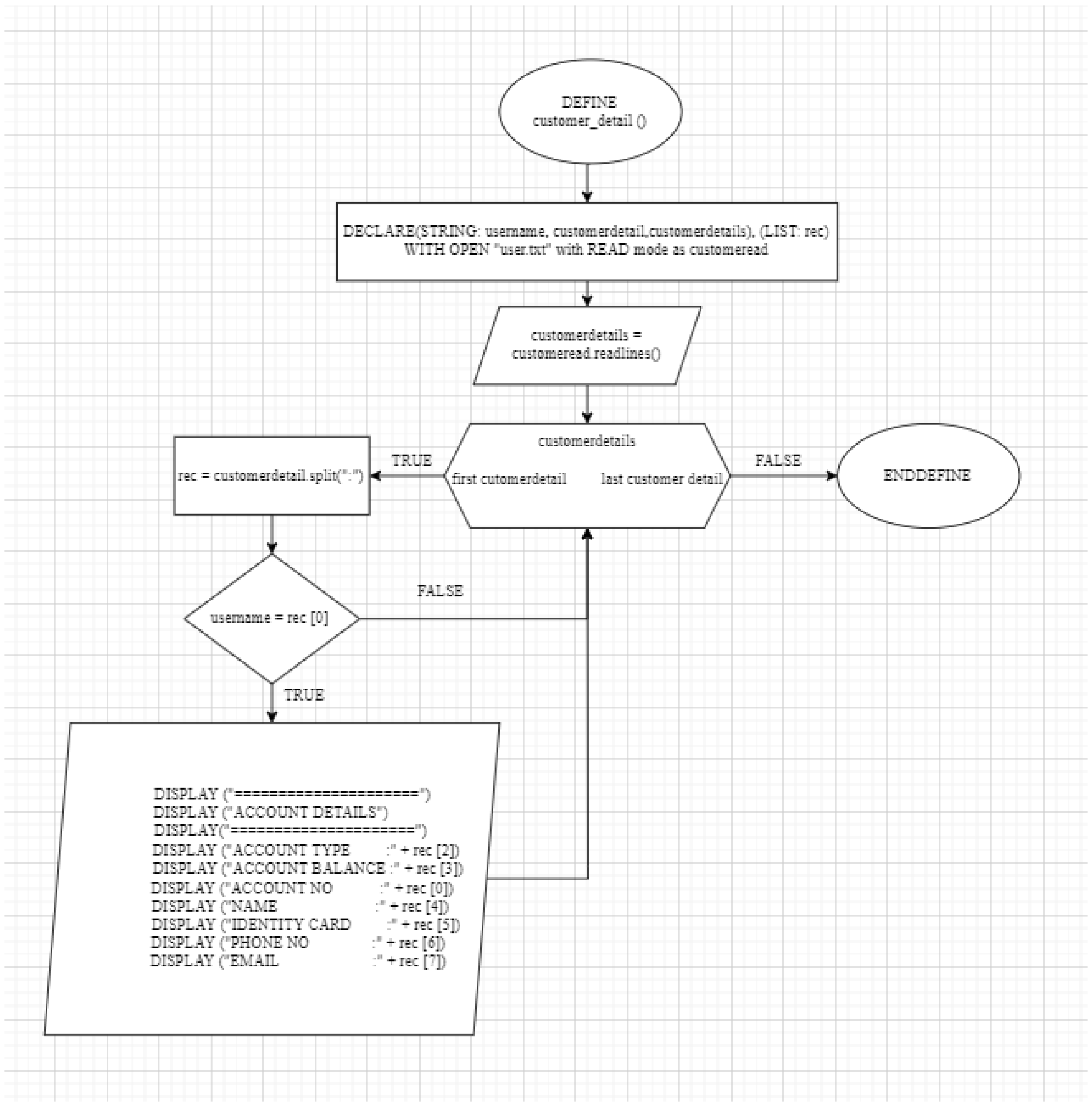
Auto generate staff id


gen_new_staffid ()
DECLARE(STRING: lines, line, default_no, no, nextid), (INTEGER: i)
i=1
SET default_no = "00000"
WITH OPEN "user.txt" with READ mode as cus_no:
lines = cus_no.readlines
line
first line
last line
True
FALSE
line [:5] = "STAFF"
True
FALSE
i = i + 1
no = CONVERT i to STRING
nextid = "STAFF" + default_no [: LENGTH of default_no -LENGTH of no] + no
RETURN nextid
ENDDEFINE

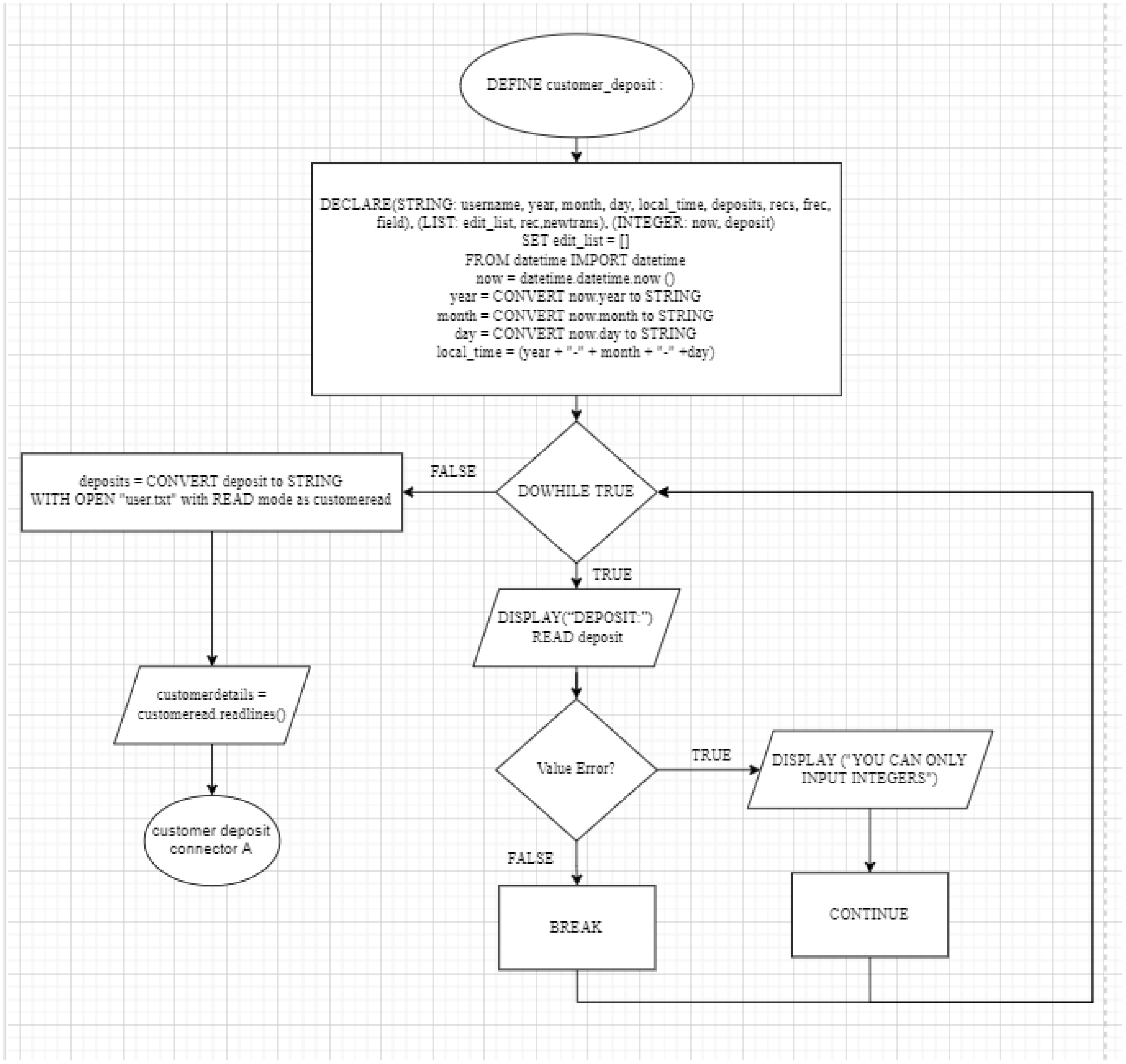
Staff and customer modify password

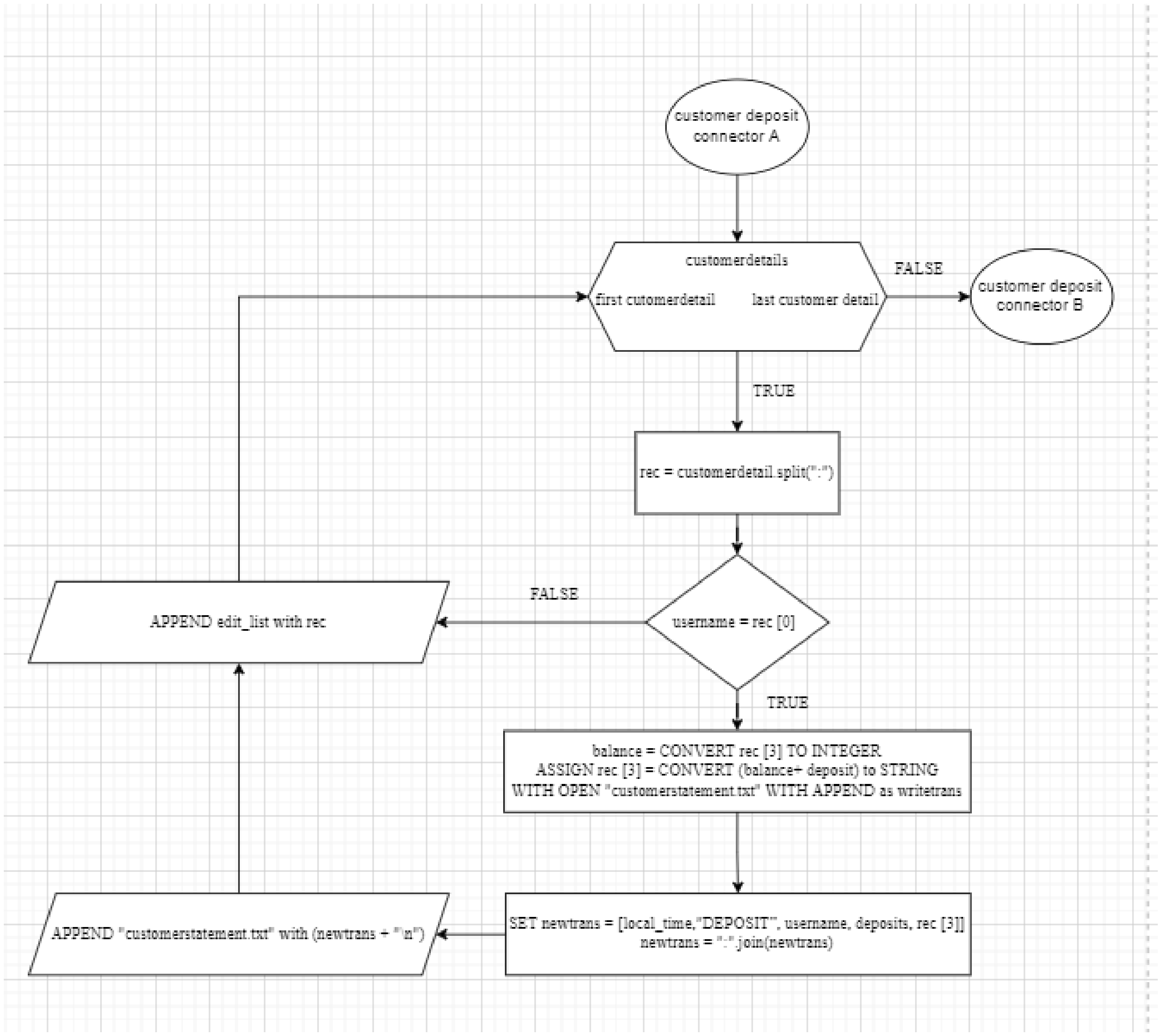

DEFINE change password()
DECLARE (STRING:userdetails, userdetail, username, recs, frec, field, newpassword),
(LIST: edit_list, rec)
WITH OPEN "user.txt" with Read mode as read
userdetails = read. readlines ()
userdetails
first userdetail
last userdetail
TRUE
FALSE
rec = userdetail. split (":")
WITH OPEN"user.txt" with WRITE mode as edit
username ==rec [0]
FALSE
APPEND edit_list with rec
TRUE
DISPLAY ("NEW PASSWORD:")
READ newpassword
ASSIGN rec [1] = newpassword
edit_list
first recs
last recs
FALSE
ENDDEFINE
TRUE
DECLARE frec = ""
WRITE ":" with frec.strip
recs
first field
last field
FALSE
TRUE
frec += field +":"

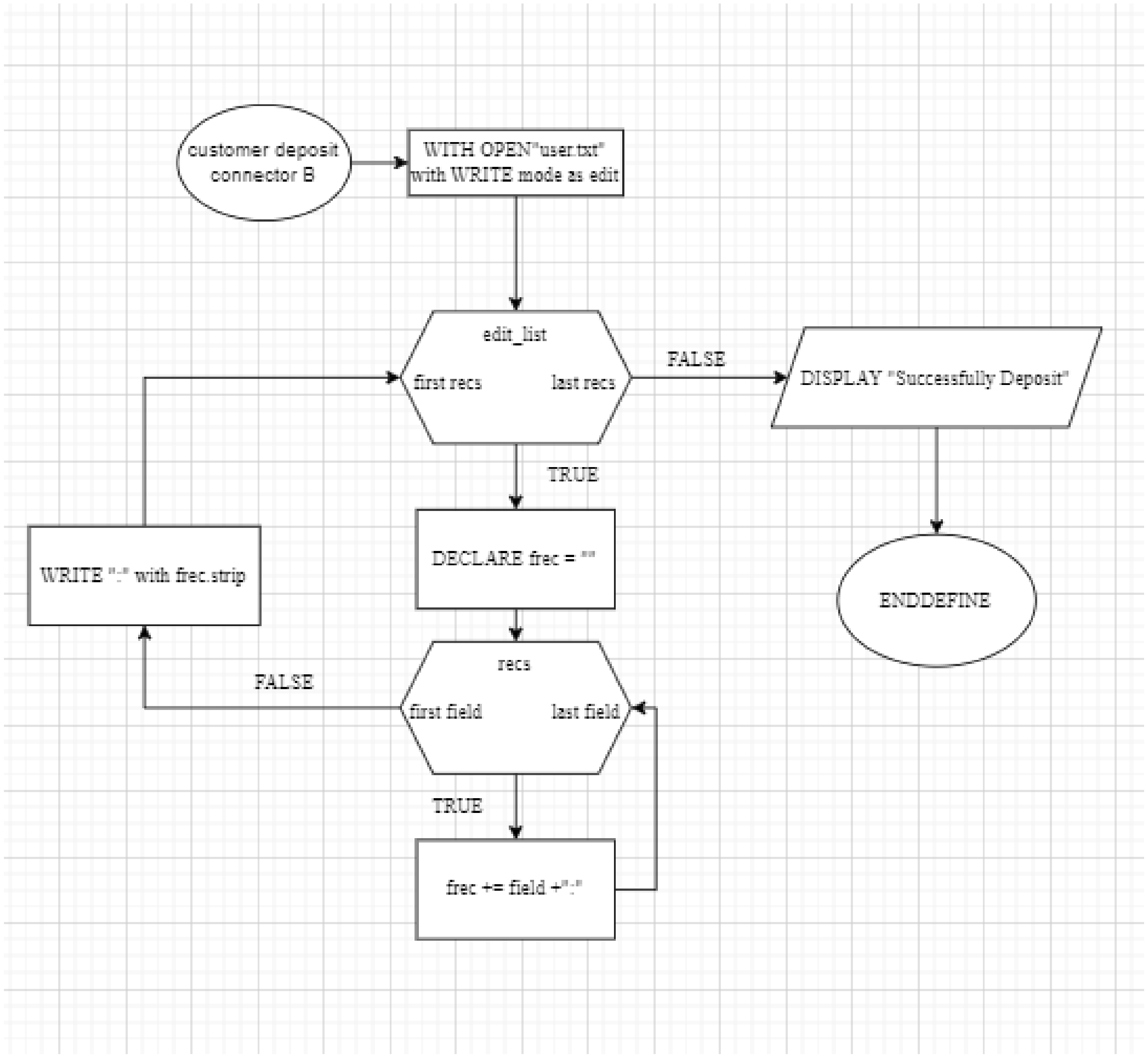
Customer view details


DEFINE customer_detail ()
DECLARE(STRING: username, customerdetail,customerdetails), (LIST: rec)
WITH OPEN "user.txt" with READ mode as customeread
customerdetails = customeread.readlines()
customerdetails
first cutomerdetail
last customer detail
TRUE
FALSE
ENDDEFINE
rec = customerdetail.split(":")
username = rec [0]
FALSE
TRUE
DISPLAY ("=====================")
DISPLAY ("ACCOUNT DETAILS")
DISPLAY("=====================")
DISPLAY ("ACCOUNT TYPE :" + rec [2])
DISPLAY ("ACCOUNT BALANCE :" + rec [3])
DISPLAY ("ACCOUNT NO :" + rec [0])
DISPLAY ("NAME :" + rec [4])
DISPLAY ("IDENTITY CARD :" + rec [5])
DISPLAY ("PHONE NO :" + rec [6])
DISPLAY ("EMAIL :" + rec [7])

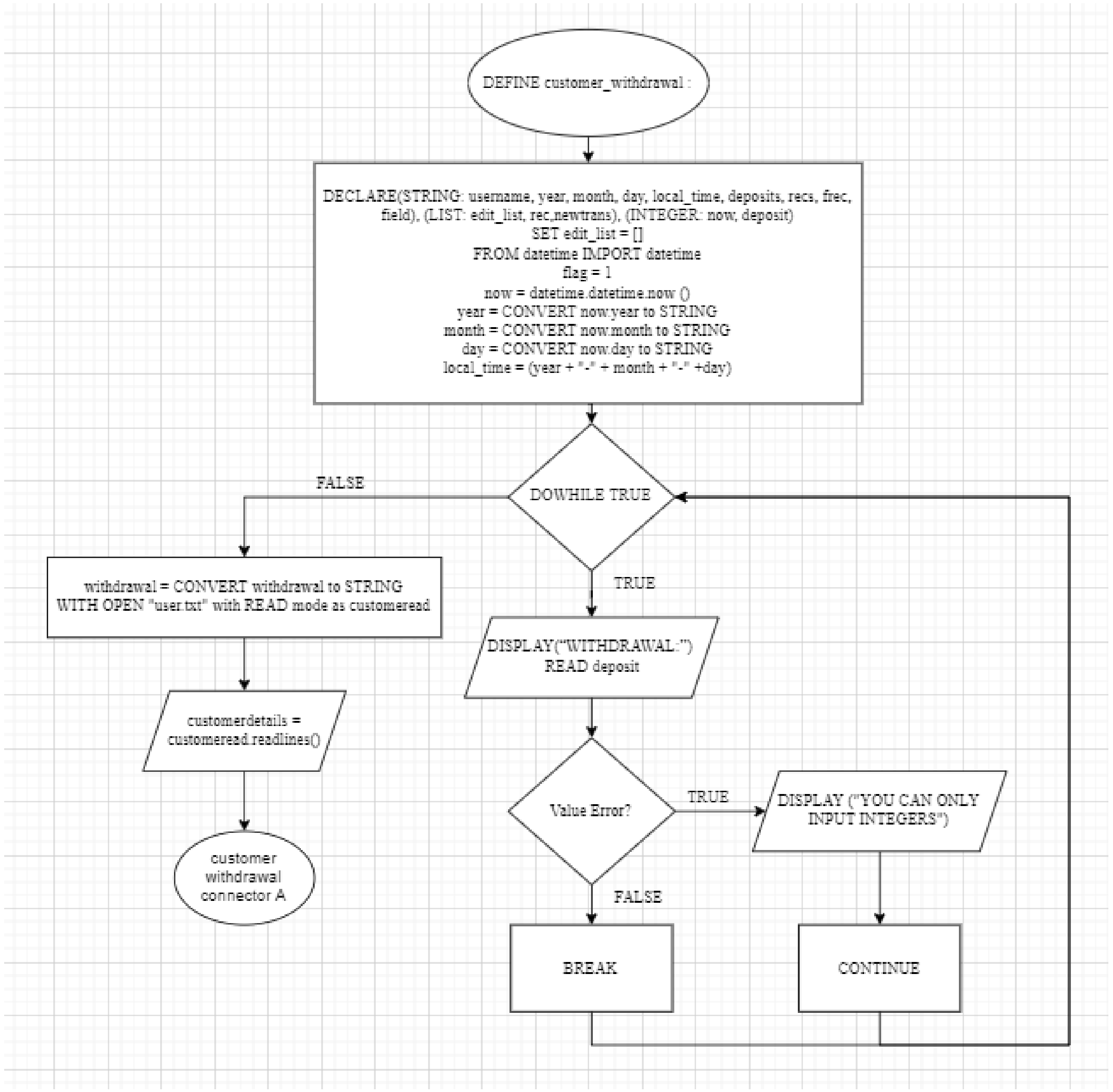
Customer deposit

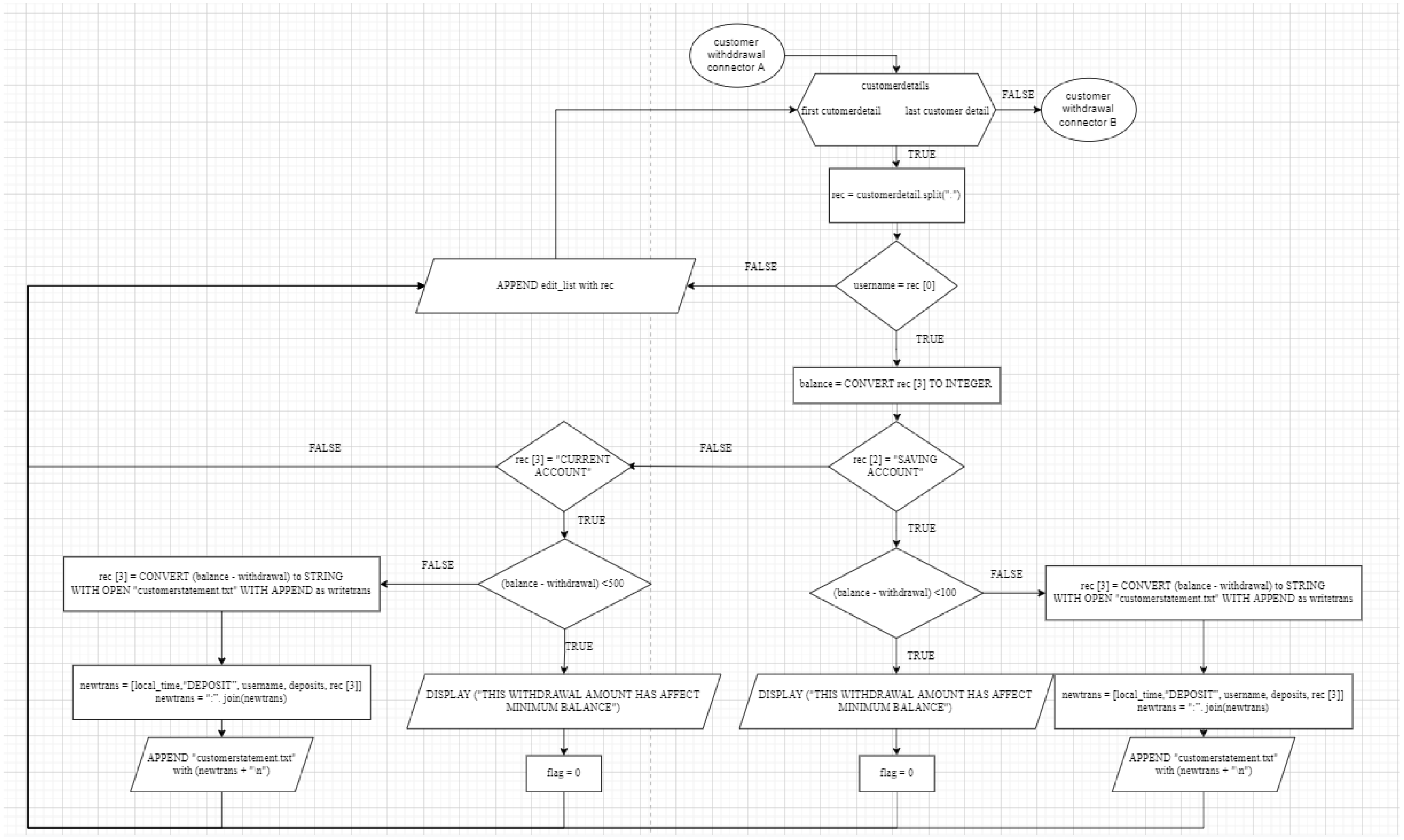

DEFINE customer_deposit :
DECLARE(STRING: username, year, month, day, local_time, deposits, recs, frec, field), (LIST: edit_list, rec,newtrans), (INTEGER: now, deposit)
SET edit_list = []
FROM datetime IMPORT datetime
now = datetime.datetime.now ()
year = CONVERT now.year to STRING
month = CONVERT now.month to STRING
day = CONVERT now.day to STRING
local_time = (year + "-" + month + "-" +day)
DOWHILE TRUE
FALSE
deposits = CONVERT deposit to STRING
WITH OPEN "user.txt" with READ mode as customeread
TRUE
DISPLAY("DEPOSIT:")
READ deposit
customerdetails = customeread.readlines()
customer deposit connector A
Value Error?
TRUE
DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
FALSE
BREAK
CONTINUE

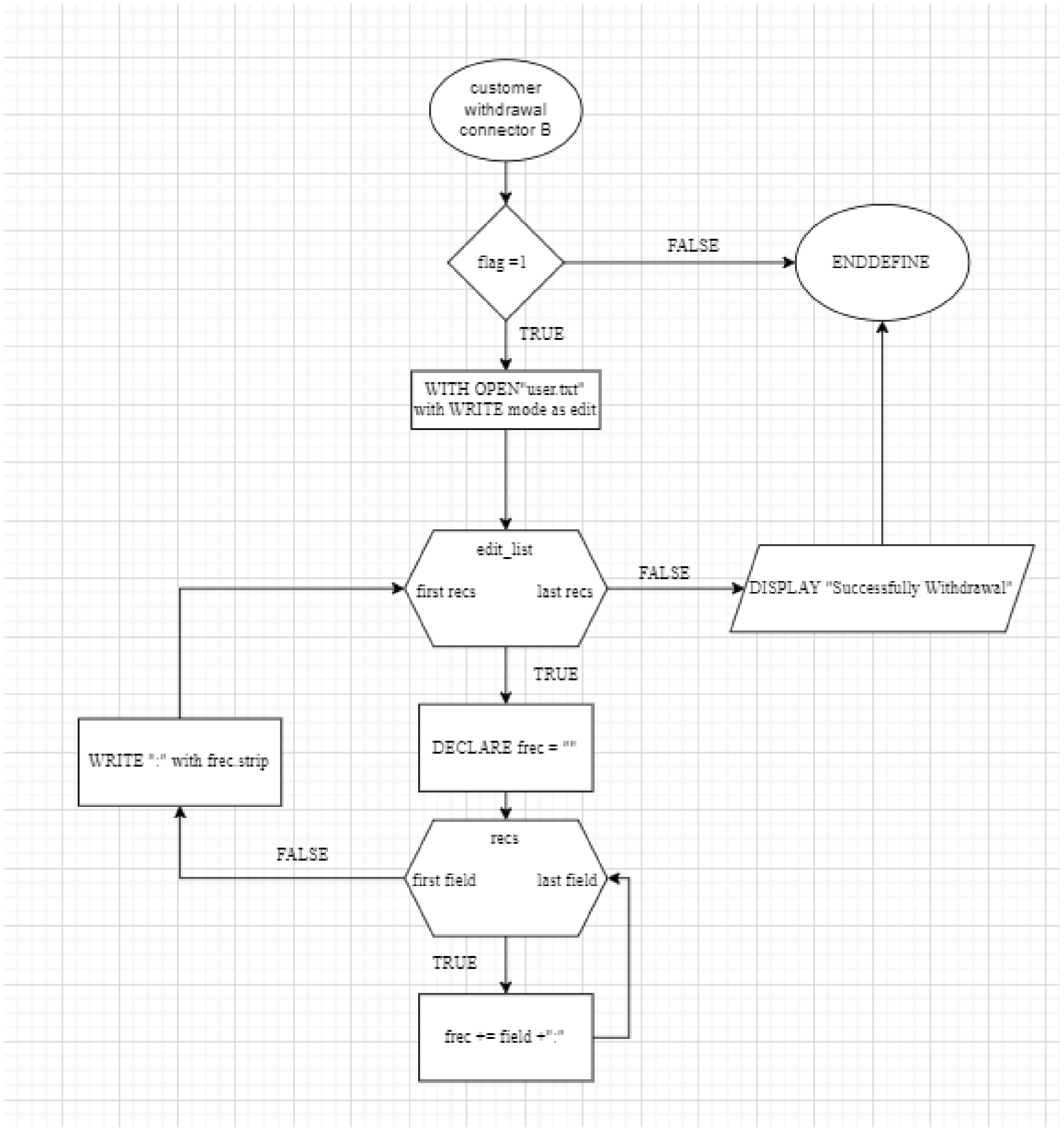

customer deposit connector A
customerdetails
first cutomerdetail
last customer detail
FALSE
customer deposit connector B
TRUE
rec = customerdetail.split(":")
FALSE
username = rec [0]
APPEND edit_list with rec
TRUE
balance = CONVERT rec [3] TO INTEGER
ASSIGN rec [3] = CONVERT (balance+ deposit) to STRING
WITH OPEN "customerstatement.txt" WITH APPEND as writetrans
SET newtrans = [local_time,"DEPOSIT", username, deposits, rec [3]]
newtrans = ":".join(newtrans)
APPEND "customerstatement.txt" with (newtrans + "\n")


customer deposit connector B
WITH OPEN"user.txt" with WRITE mode as edit
edit_list
first recs
last recs
FALSE
DISPLAY "Successfully Deposit"
TRUE
DECLARE frec = ""
WRITE ":" with frec.strip
ENDDEFINE
recs
FALSE
first field
last field
TRUE
frec += field +":"

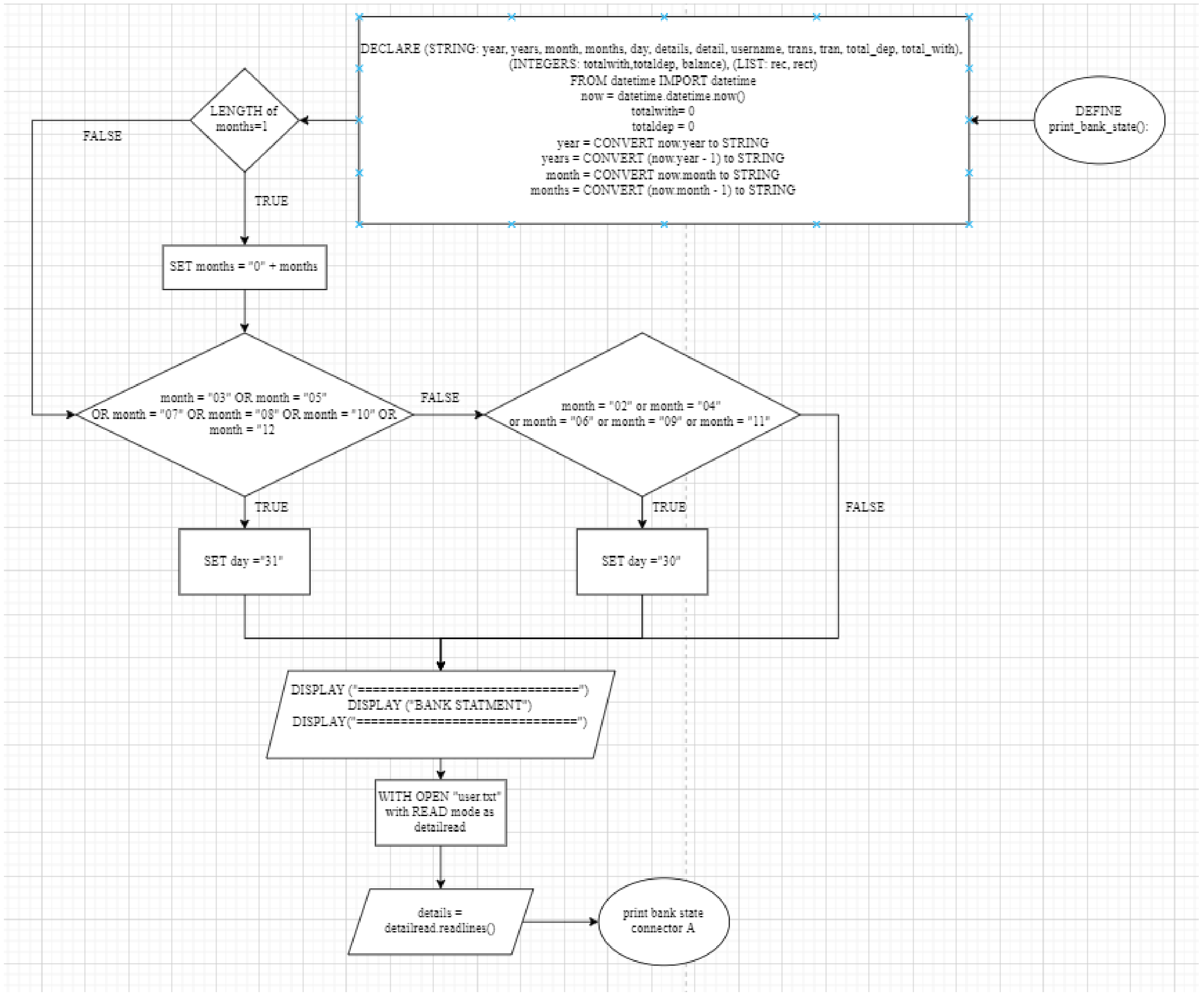
Customer withdrawal

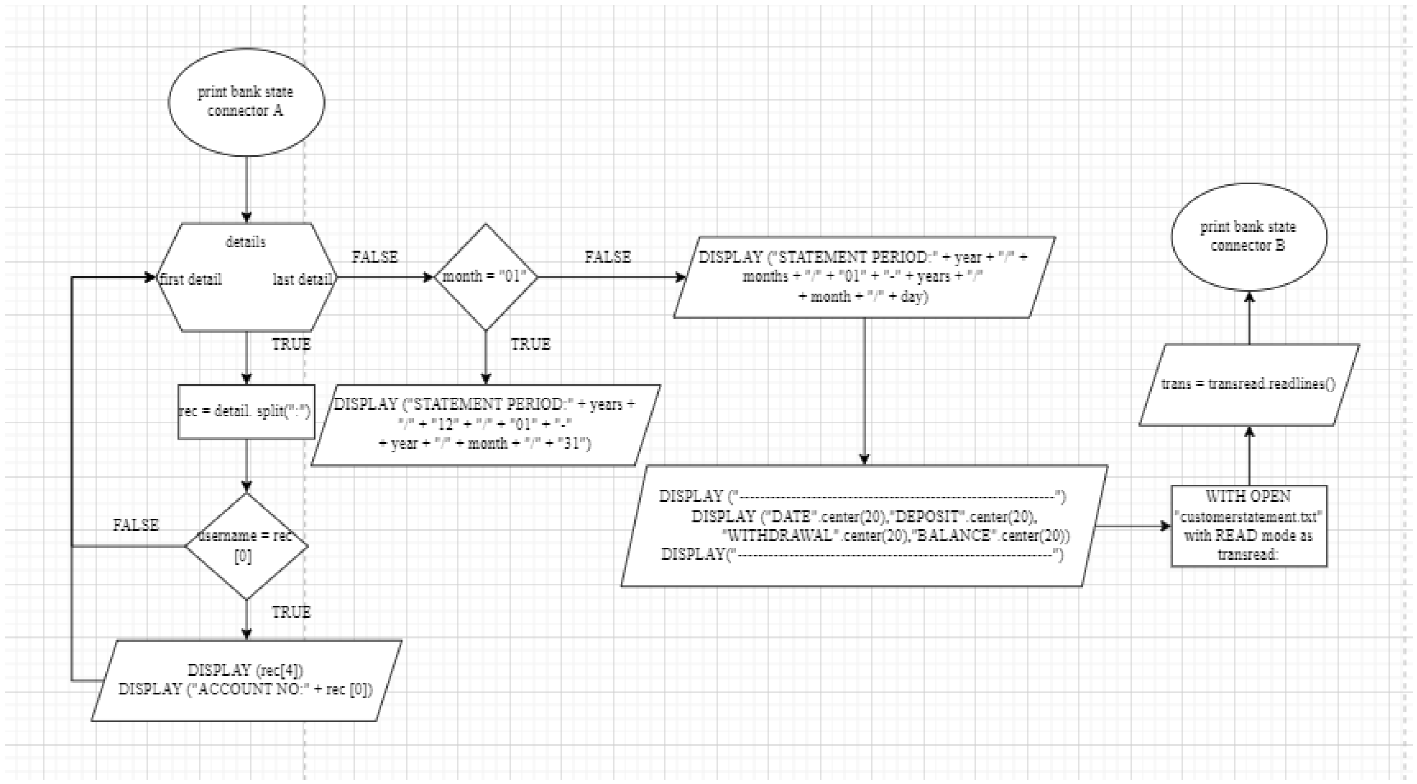

DEFINE customer_withdrawal :
DECLARE(STRING: username, year, month, day, local_time, deposits, recs, frec, field), (LIST: edit_list, rec,newtrans), (INTEGER: now, deposit)
SET edit_list = []
FROM datetime IMPORT datetime
flag = 1
now = datetime.datetime.now ()
year = CONVERT now.year to STRING
month = CONVERT now.month to STRING
day = CONVERT now.day to STRING
local_time = (year + "-" + month + "-" +day)
DOWHILE TRUE
FALSE
TRUE
withdrawal = CONVERT withdrawal to STRING
WITH OPEN "user.txt" with READ mode as customeread
DISPLAY("WITHDRAWAL:")
READ deposit
customerdetails = customeread.readlines()
Value Error?
TRUE
DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
FALSE
customer withdrawal connector A
BREAK
CONTINUE

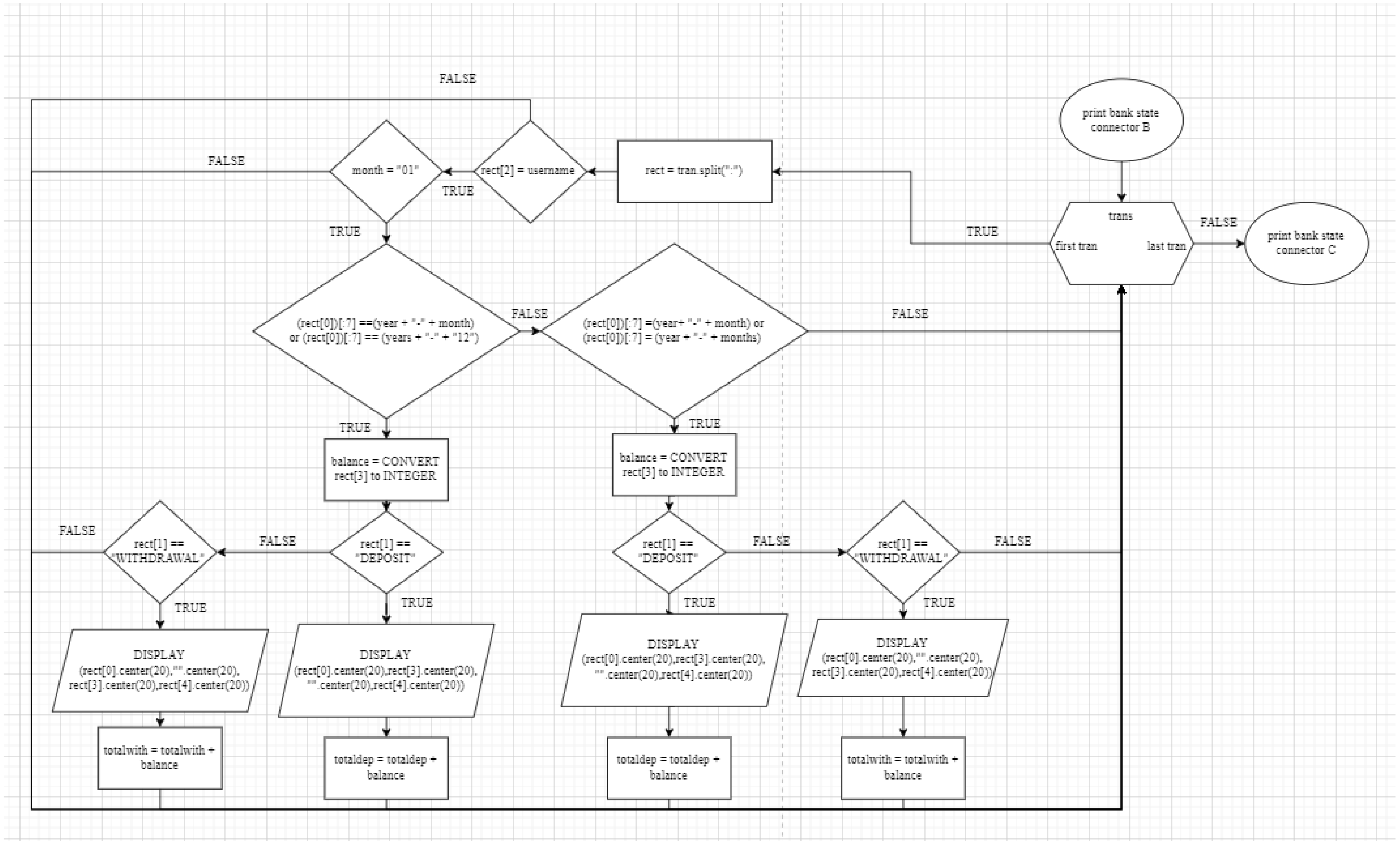

customer withddrawal connector A
customerdetails
first cutomerdetail
last customer detail
FALSE
customer withdrawal connector B
TRUE
rec = customerdetail.split(":")
username = rec [0]
FALSE
APPEND edit_list with rec
TRUE
balance = CONVERT rec [3] TO INTEGER
rec [2] = "SAVING ACCOUNT"
FALSE
rec [3] = "CURRENT ACCOUNT"
FALSE
TRUE
(balance - withdrawal) <500
FALSE
rec [3] = CONVERT (balance - withdrawal) to STRING WITH OPEN "customerstatement.txt" WITH APPEND as writetrans
newtrans = [local_time,"DEPOSIT", username, deposits, rec [3]] newtrans = ":". join(newtrans)
APPEND "customerstatement.txt" with (newtrans + "\n")
TRUE
DISPLAY ("THIS WITHDRAWAL AMOUNT HAS AFFECT MINIMUM BALANCE")
flag = 0
TRUE
(balance - withdrawal) <100
FALSE
rec [3] = CONVERT (balance - withdrawal) to STRING WITH OPEN "customerstatement.txt" WITH APPEND as writetrans
TRUE
DISPLAY ("THIS WITHDRAWAL AMOUNT HAS AFFECT MINIMUM BALANCE")
flag = 0
newtrans = [local_time,"DEPOSIT", username, deposits, rec [3]] newtrans = ":". join(newtrans)
APPEND "customerstatement.txt" with (newtrans + "\n")

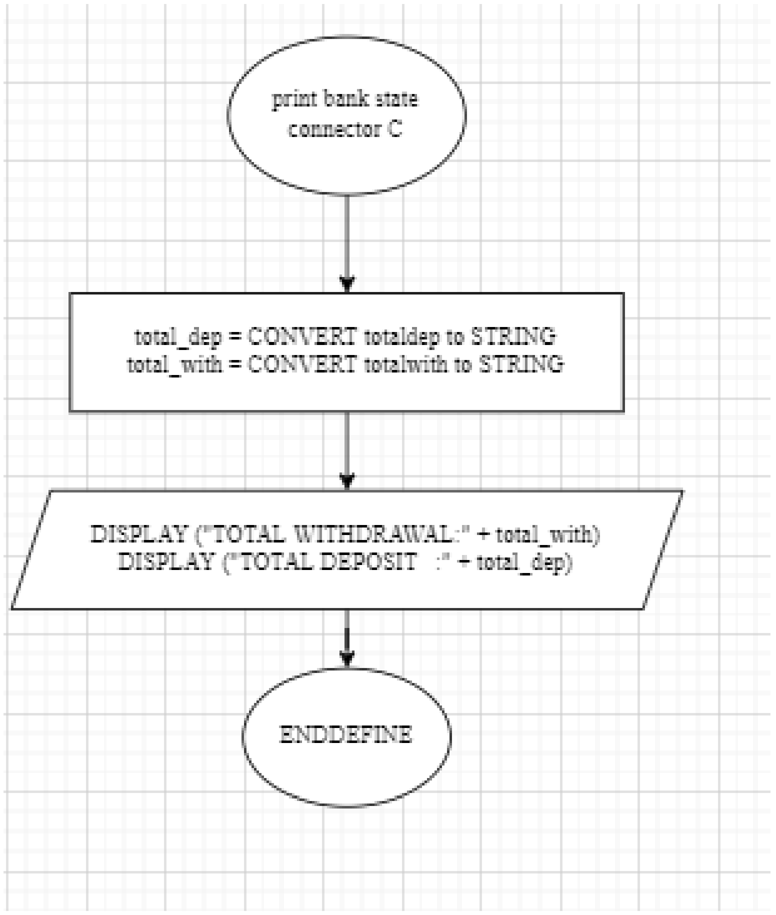

customer withdrawal connector B
flag =1
FALSE
ENDDEFINE
TRUE
WITH OPEN"user.txt" with WRITE mode as edit
edit_list
first recs
last recs
FALSE
DISPLAY "Successfully Withdrawal"
TRUE
DECLARE frec = ""
WRITE ":" with frec.strip
recs
first field
last field
FALSE
TRUE
frec += field +":"

Customer print bank statement

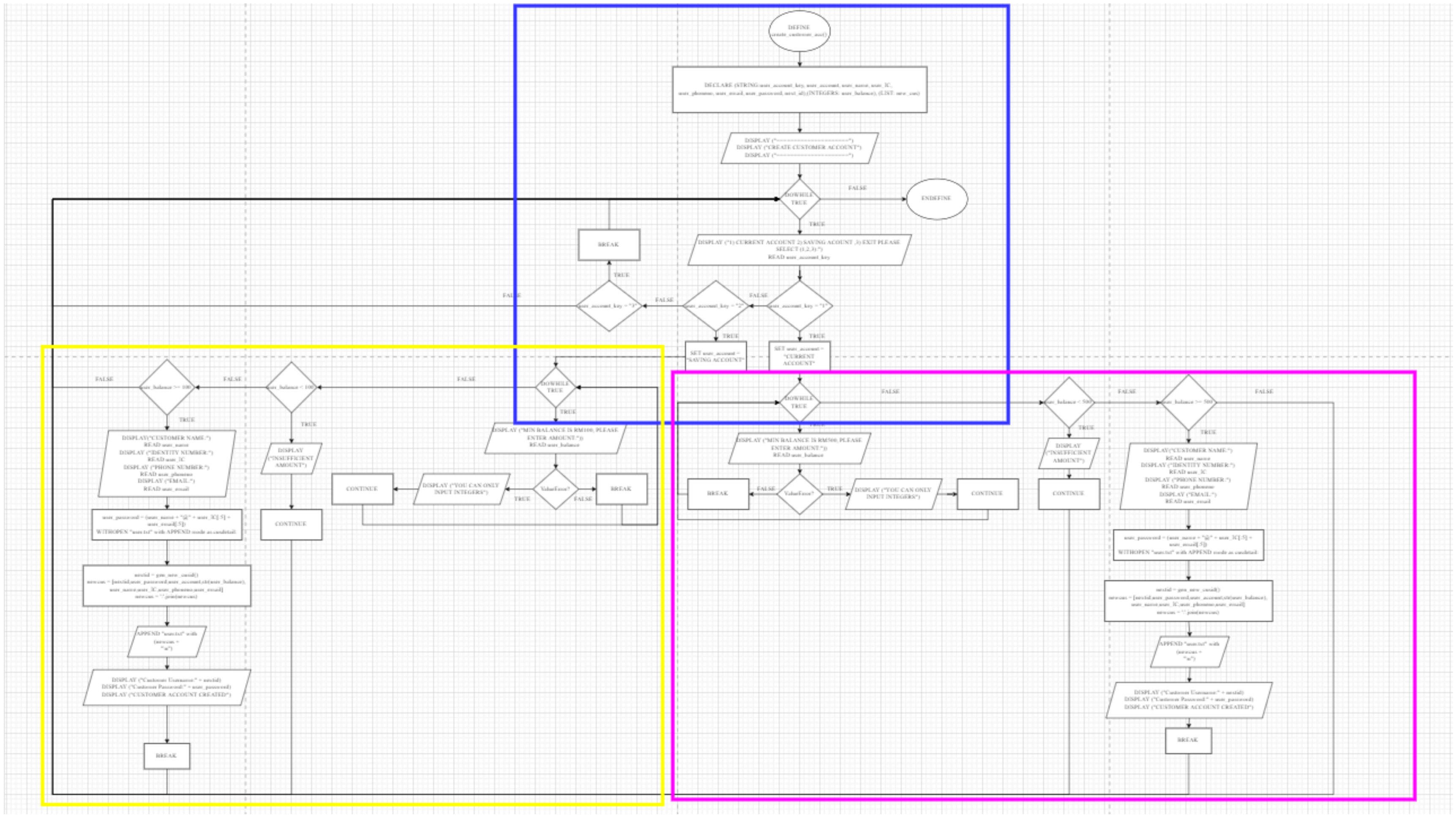

DEFINE print_bank_state():
DECLARE (STRING: year, years, month, months, day, details, detail, username, trans, tran, total_dep, total_with), (INTEGERS: totalwith,totaldep, balance), (LIST: rec, rect)
FROM datetime IMPORT datetime
now = datetime.datetime.now()
totalwith= 0
totaldep = 0
year = CONVERT now.year to STRING
years = CONVERT (now.year - 1) to STRING
month = CONVERT now.month to STRING
months = CONVERT (now.month - 1) to STRING
LENGTH of months=1
FALSE
TRUE
SET months = "0" + months
month = "03" OR month = "05" OR month = "07" OR month = "08" OR month = "10" OR month = "12
FALSE
month = "02" or month = "04" or month = "06" or month = "09" or month = "11"
TRUE
SET day ="31"
TRUE
SET day ="30"
FALSE
DISPLAY ("==============================")
DISPLAY ("BANK STATMENT")
DISPLAY("==============================")
WITH OPEN "user.txt" with READ mode as detailread
details = detailread.readlines()
print bank state connector A

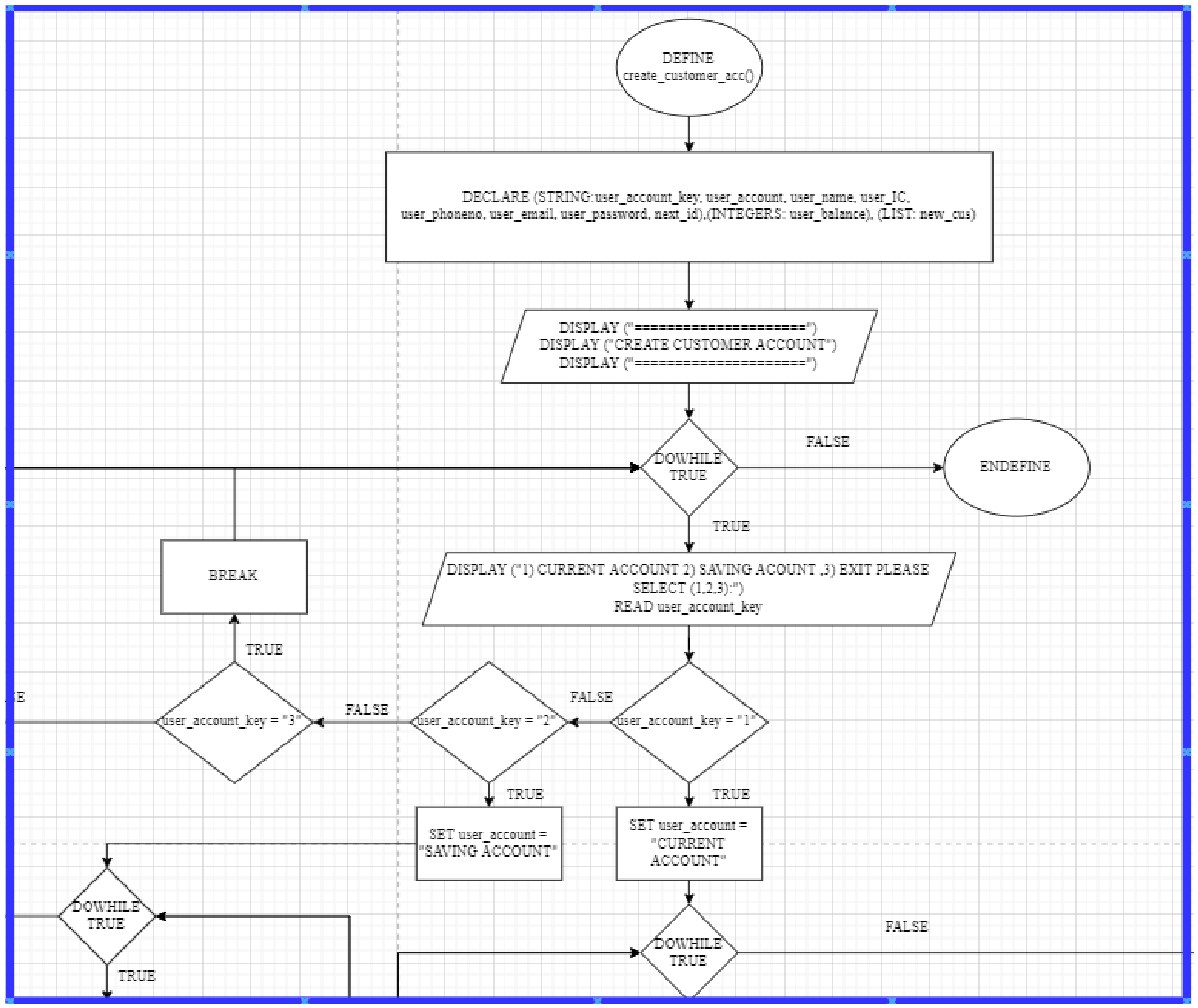

print bank state connector A
details
first detail
last detail
FALSE
month = "01"
FALSE
DISPLAY ("STATEMENT PERIOD:" + year + "/" + months + "/" + "01" + "-" + years + "/" + month + "/" + day)
TRUE
TRUE
rec = detail. split(":")
DISPLAY ("STATEMENT PERIOD:" + years + "/" + "12" + "/" + "01" + "-" + year + "/" + month + "/" + "31")
FALSE
username = rec [0]
TRUE
DISPLAY (rec[4])
DISPLAY ("ACCOUNT NO:" + rec [0])
DISPLAY ("--")
DISPLAY ("DATE".center(20),"DEPOSIT".center(20), "WITHDRAWAL".center(20),"BALANCE".center(20))
DISPLAY("--")
WITH OPEN "customerstatement.txt" with READ mode as transread:
trans = transread.readlines()
print bank state connector B

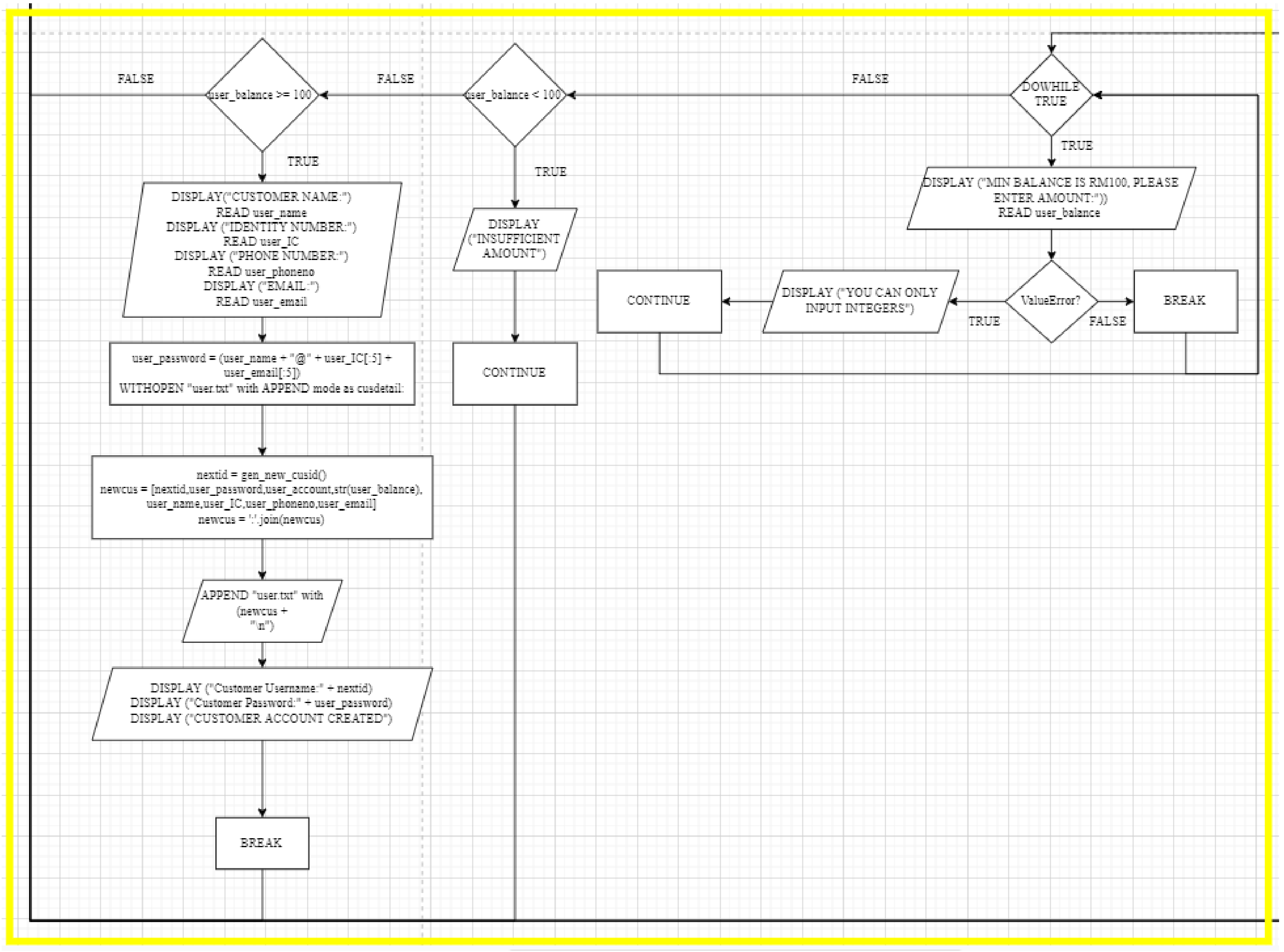

print bank state connector B
trans
first tran
last tran
FALSE
print bank state connector C
TRUE
rect = tran.split(":")
rect[2] = username
FALSE
TRUE
month = "01"
FALSE
TRUE
(rect[0])[:7] ==(year + "-" + month) or (rect[0])[:7] == (years + "-" + "12")
FALSE
(rect[0])[:7] =(year+ "-" + month) or (rect[0])[:7] = (year + "-" + months)
FALSE
TRUE
balance = CONVERT rect[3] to INTEGER
TRUE
balance = CONVERT rect[3] to INTEGER
rect[1] == "DEPOSIT"
FALSE
rect[1] == "WITHDRAWAL"
FALSE
TRUE
DISPLAY (rect[0].center(20),"".center(20), rect[3].center(20),rect[4].center(20))
totalwith = totalwith + balance
TRUE
DISPLAY (rect[0].center(20),rect[3].center(20), "".center(20),rect[4].center(20))
totaldep = totaldep + balance
rect[1] == "DEPOSIT"
FALSE
TRUE
DISPLAY (rect[0].center(20),rect[3].center(20), "".center(20),rect[4].center(20))
totaldep = totaldep + balance
rect[1] == "WITHDRAWAL"
FALSE
TRUE
DISPLAY (rect[0].center(20),"".center(20), rect[3].center(20),rect[4].center(20))
totalwith = totalwith + balance

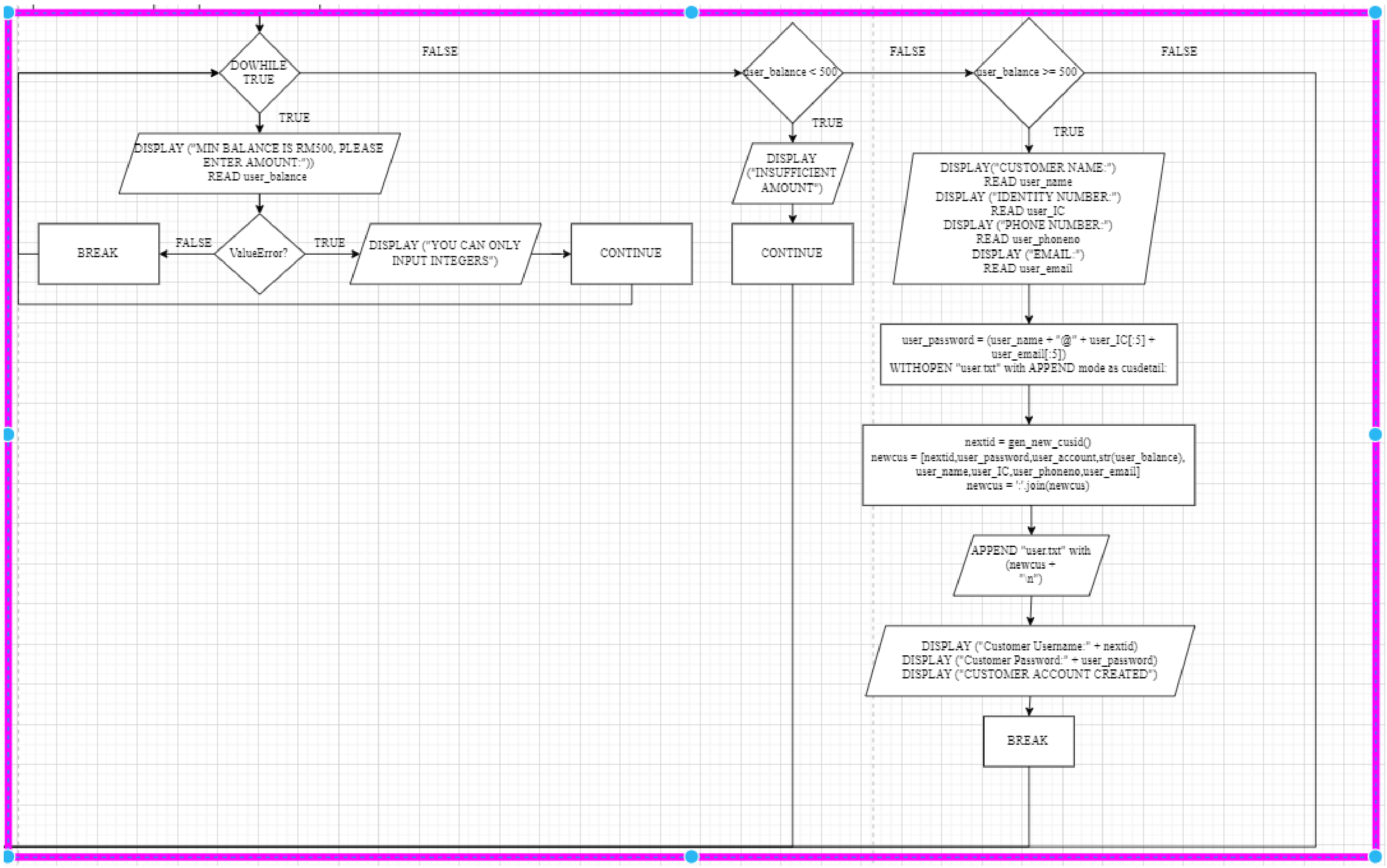

print bank state connector C
total_dep = CONVERT totaldep to STRING
total_with = CONVERT totalwith to STRING
DISPLAY ("TOTAL WITHDRAWAL:" + total_with)
DISPLAY ("TOTAL DEPOSIT   :" + total_dep)
ENDDEFINE

Staff create customer account

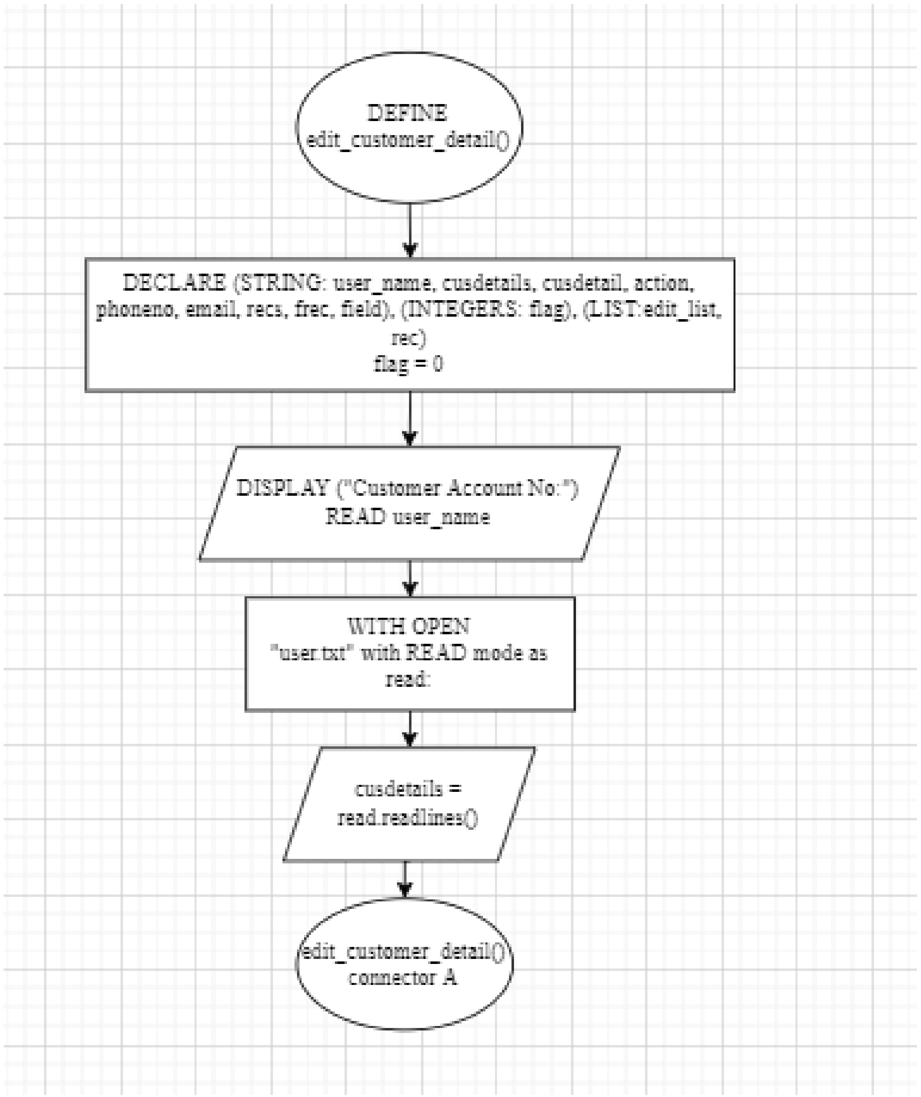


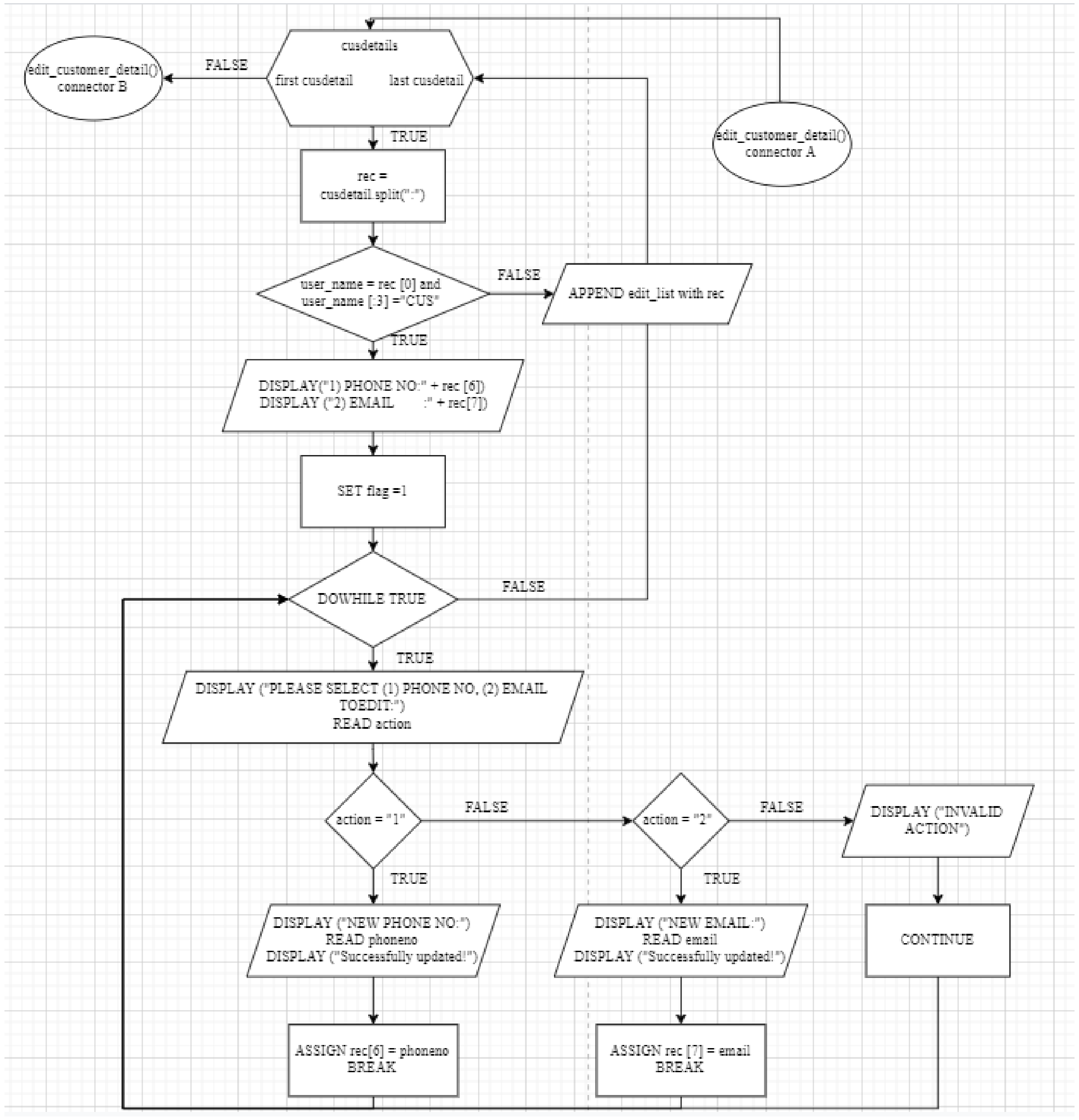

DEFINE create_customer_acc()
DECLARE (STRING:user_account_key, user_account, user_name, user_IC, user_phoneno, user_email, user_password, next_id),(INTEGERS: user_balance), (LIST: new_cus)
DISPLAY ("====================")
DISPLAY ("CREATE CUSTOMER ACCOUNT")
DISPLAY ("====================")
DOWHILE TRUE
FALSE
ENDEFINE
TRUE
DISPLAY ("1) CURRENT ACCOUNT 2) SAVING ACOUNT ,3) EXIT PLEASE SELECT (1,2,3):")
READ user_account_key
BREAK
TRUE
user_account_key = "3"
FALSE
user_account_key = "2"
FALSE
user_account_key = "1"
TRUE
TRUE
SET user_account = "SAVING ACCOUNT"
SET user_account = "CURRENT ACCOUNT"
DOWHILE TRUE
TRUE
DOWHILE TRUE
FALSE

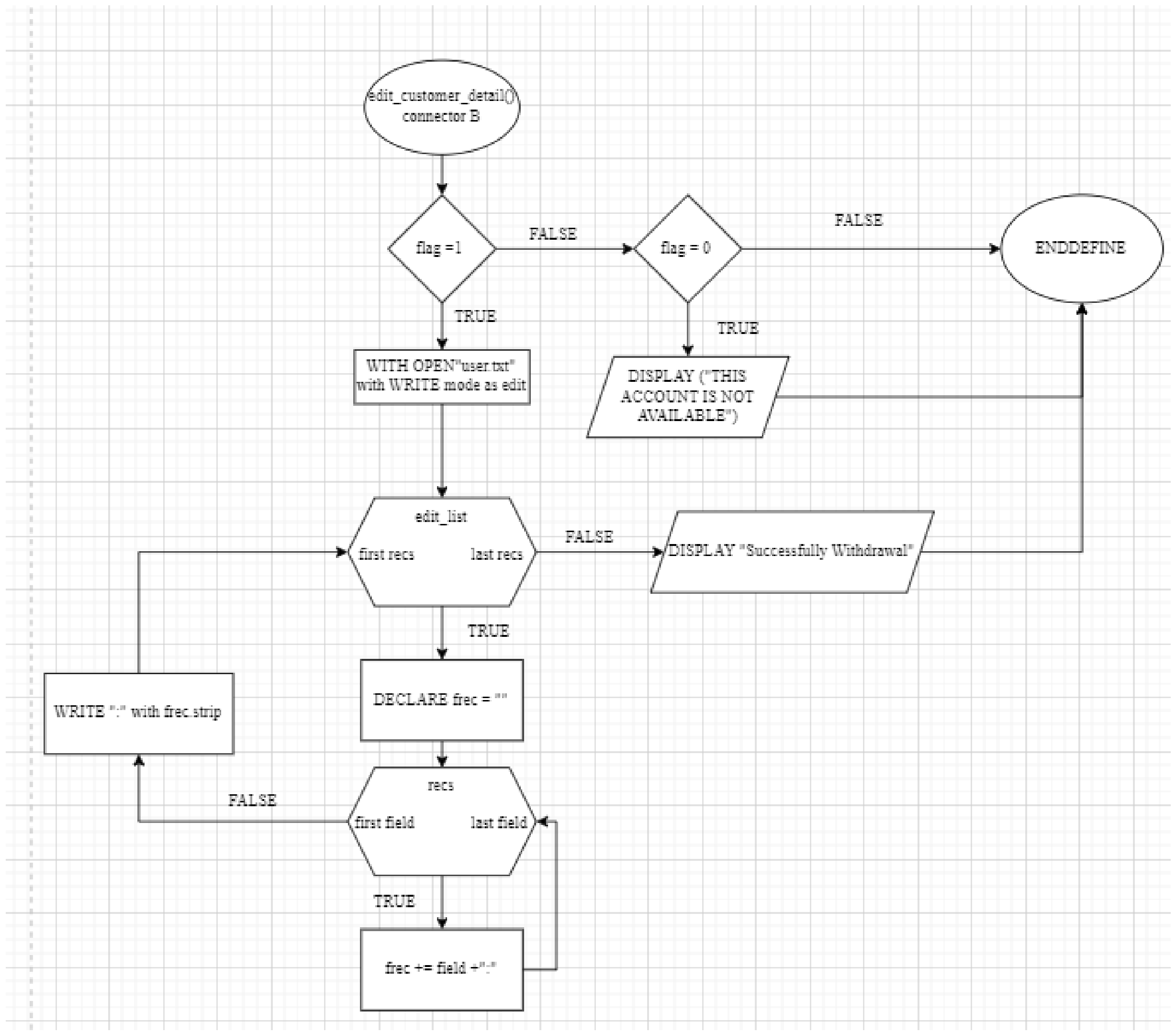

FALSE
user_balance >= 100
FALSE
user_balance < 100
FALSE
DOWHILE TRUE
TRUE
TRUE
TRUE
DISPLAY("CUSTOMER NAME:") READ user_name DISPLAY ("IDENTITY NUMBER:") READ user_IC DISPLAY ("PHONE NUMBER:") READ user_phoneno DISPLAY ("EMAIL:") READ user_email
DISPLAY ("INSUFFICIENT AMOUNT")
DISPLAY ("MIN BALANCE IS RM100, PLEASE ENTER AMOUNT:")) READ user_balance
CONTINUE
DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
ValueError?
TRUE
FALSE
BREAK
user_password = (user_name + "@" + user_IC[:5] + user_email[:5]) WITHOPEN "user.txt" with APPEND mode as cusdetail:
CONTINUE
nextid = gen_new_cusid() newcus = [nextid,user_password,user_account,str(user_balance), user_name,user_IC,user_phoneno,user_email] newcus = ':'.join(newcus)
APPEND "user.txt" with (newcus + "\n")
DISPLAY ("Customer Username:" + nextid) DISPLAY ("Customer Password:" + user_password) DISPLAY ("CUSTOMER ACCOUNT CREATED")
BREAK


DOWHILE TRUE
FALSE
TRUE
DISPLAY ("MIN BALANCE IS RM500, PLEASE ENTER AMOUNT:"))
READ user_balance
ValueError?
FALSE
TRUE
BREAK
DISPLAY ("YOU CAN ONLY INPUT INTEGERS")
CONTINUE
user_balance < 500
FALSE
TRUE
DISPLAY ("INSUFFICIENT AMOUNT")
CONTINUE
user_balance >= 500
FALSE
TRUE
DISPLAY("CUSTOMER NAME:")
READ user_name
DISPLAY ("IDENTITY NUMBER:")
READ user_IC
DISPLAY ("PHONE NUMBER:")
READ user_phoneno
DISPLAY ("EMAIL:")
READ user_email
user_password = (user_name + "@" + user_IC[:5] + user_email[:5])
WITHOPEN "user.txt" with APPEND mode as cusdetail:
nextid = gen_new_cusid()
newcus = [nextid,user_password,user_account,str(user_balance), user_name,user_IC,user_phoneno,user_email]
newcus = ':'.join(newcus)
APPEND "user.txt" with (newcus + "\n")
DISPLAY ("Customer Username:" + nextid)
DISPLAY ("Customer Password:" + user_password)
DISPLAY ("CUSTOMER ACCOUNT CREATED")
BREAK

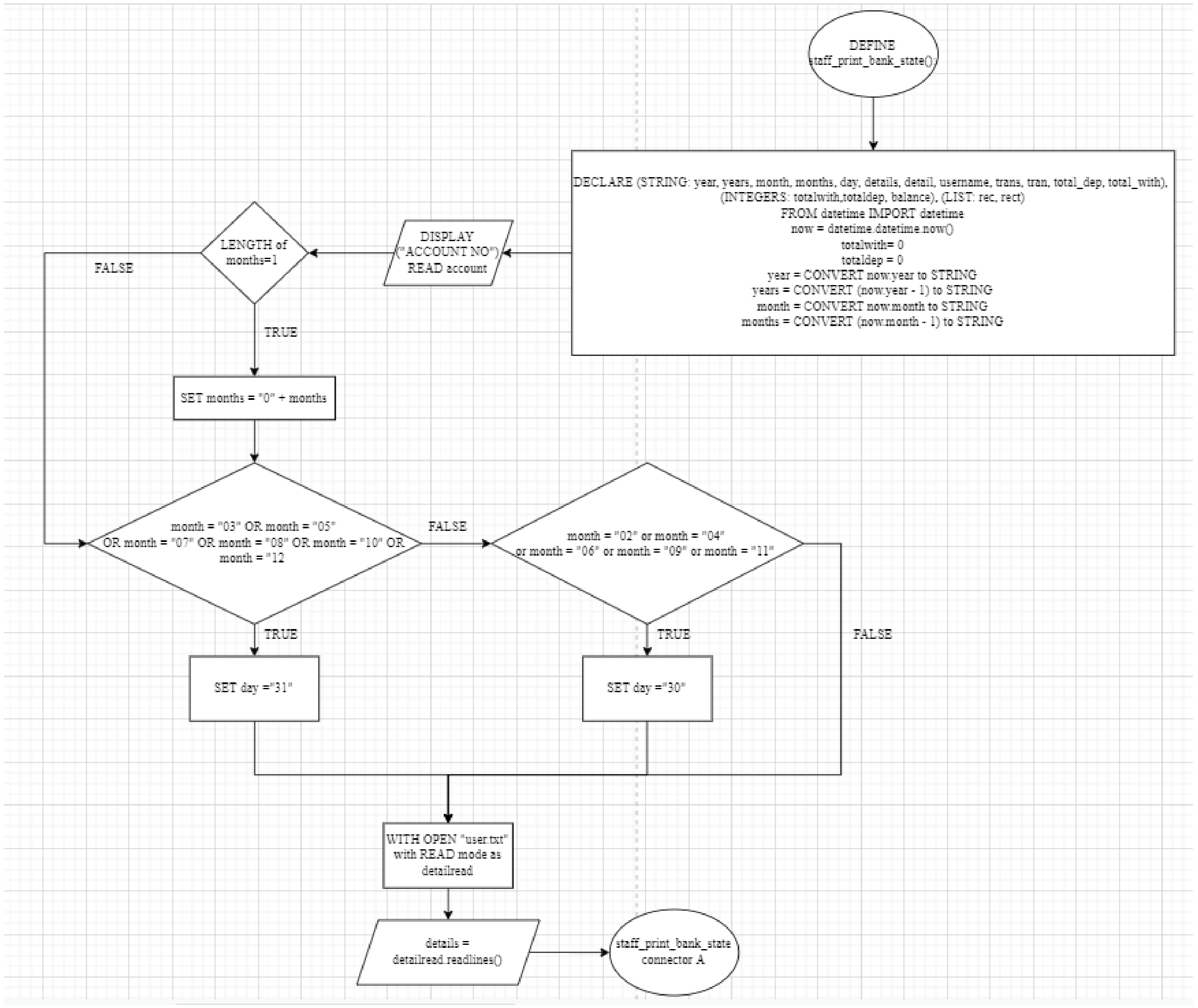
Staff edit customer details

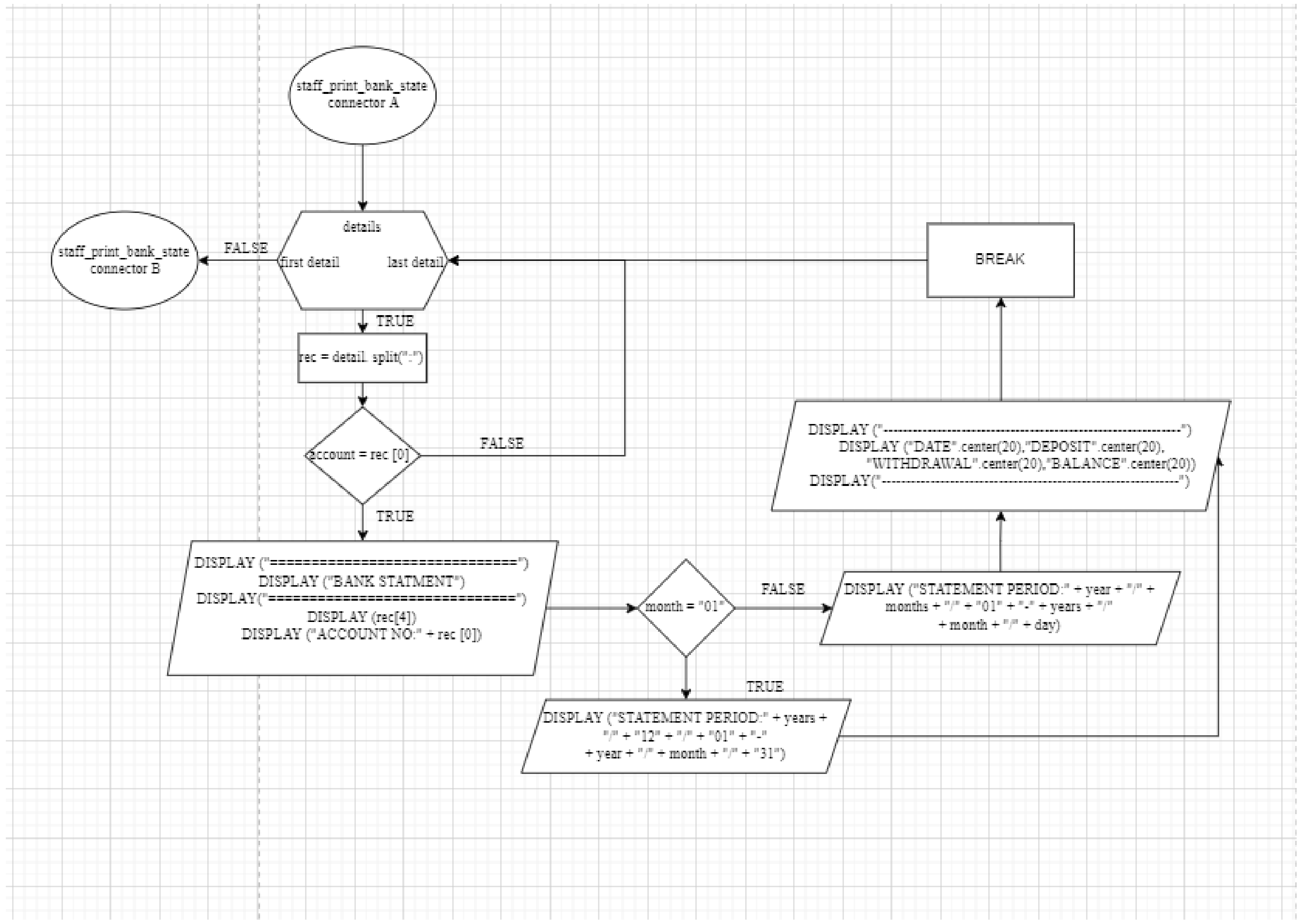

DEFINE edit_customer_detail()
DECLARE (STRING: user_name, cusdetails, cusdetail, action, phoneno, email, recs, frec, field), (INTEGERS: flag), (LIST:edit_list, rec)
flag = 0
DISPLAY ("Customer Account No:")
READ user_name
WITH OPEN "user.txt" with READ mode as read:
cusdetails = read.readlines()
edit_customer_detail() connector A

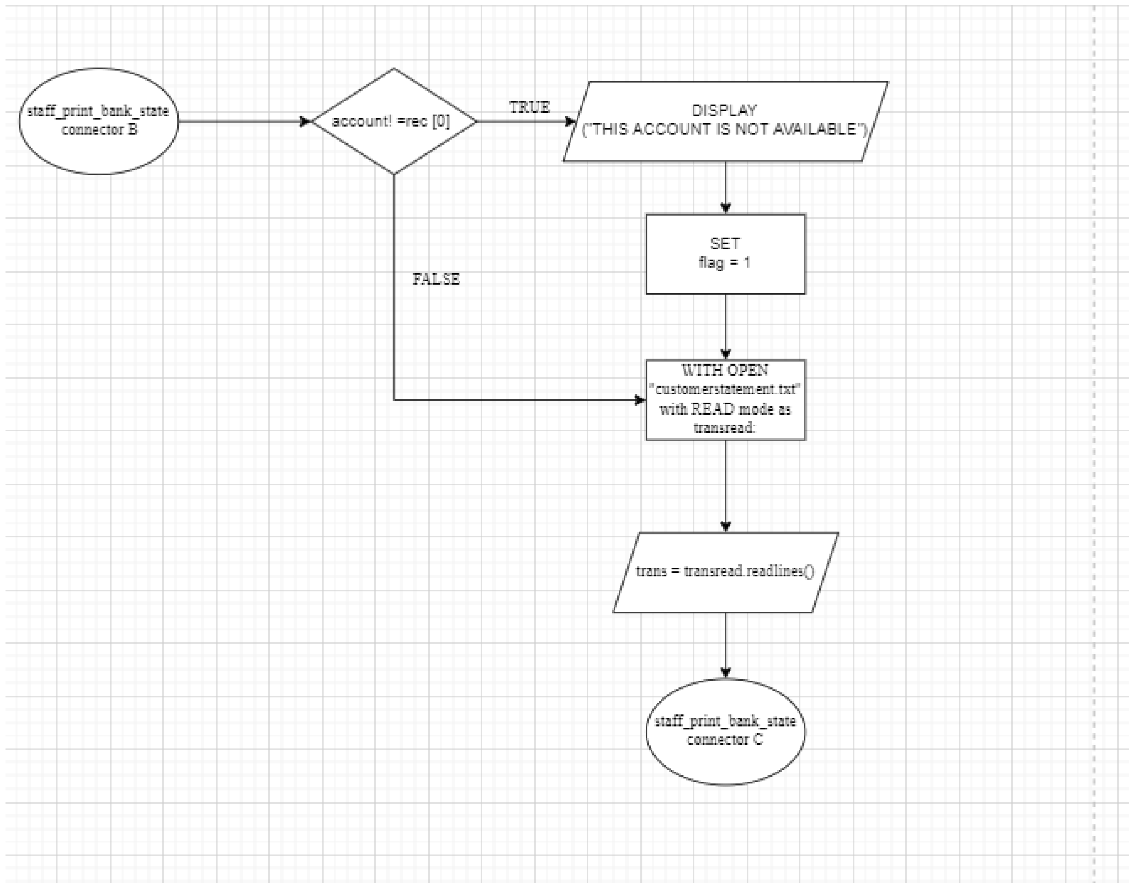

cusdetails
first cusdetail
last cusdetail
FALSE
edit_customer_detail() connector B
TRUE
rec = cusdetail.split(":")
edit_customer_detail() connector A
user_name = rec [0] and user_name [:3] ="CUS"
FALSE
APPEND edit_list with rec
TRUE
DISPLAY("1) PHONE NO:" + rec [6])
DISPLAY ("2) EMAIL :" + rec[7])
SET flag =1
DOWHILE TRUE
FALSE
TRUE
DISPLAY ("PLEASE SELECT (1) PHONE NO, (2) EMAIL TOEDIT:")
READ action
action = "1"
FALSE
action = "2"
FALSE
DISPLAY ("INVALID ACTION")
TRUE
TRUE
DISPLAY ("NEW PHONE NO:")
READ phoneno
DISPLAY ("Successfully updated!")
DISPLAY ("NEW EMAIL:")
READ email
DISPLAY ("Successfully updated!")
CONTINUE
ASSIGN rec[6] = phoneno
BREAK
ASSIGN rec [7] = email
BREAK

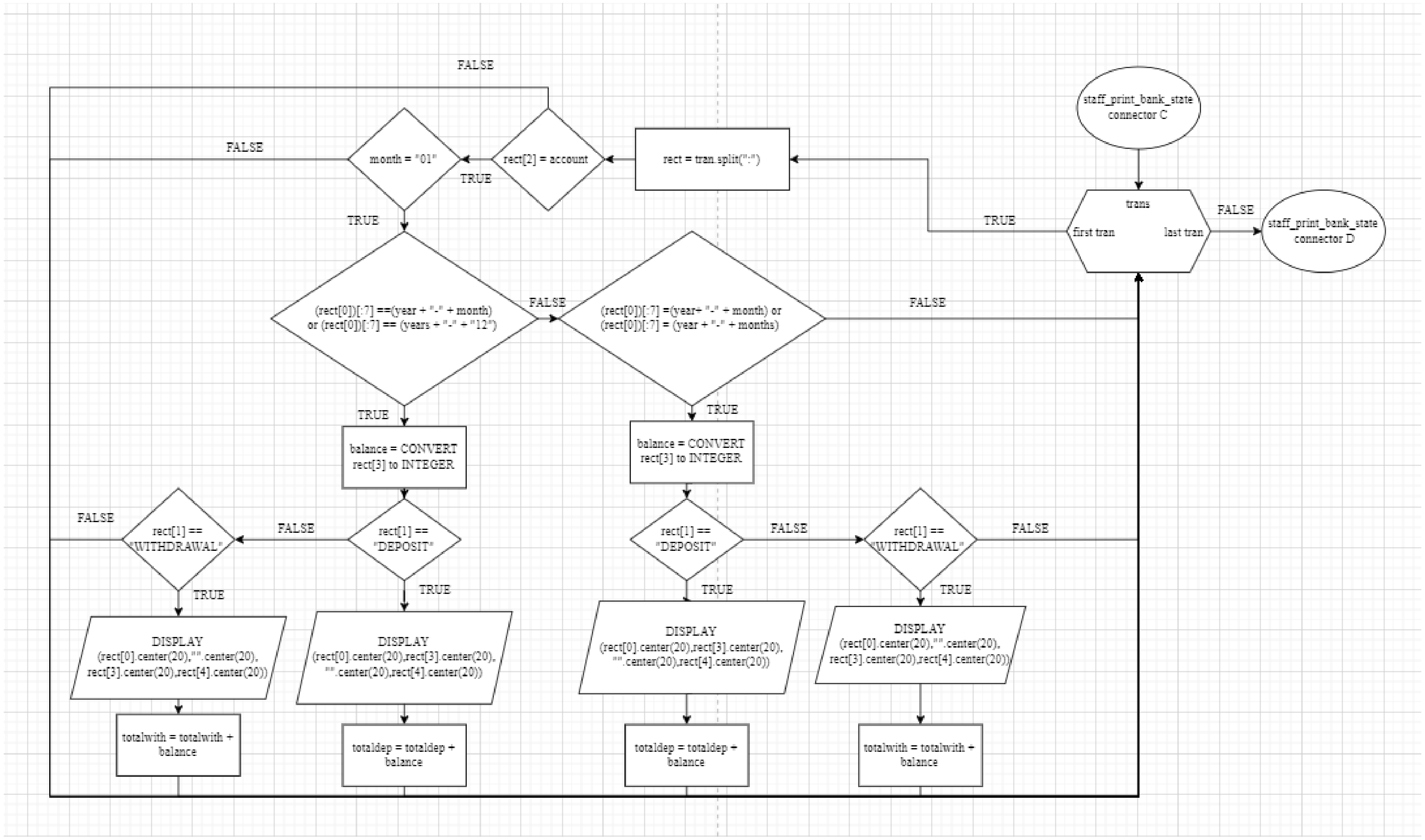

edit_customer_detail() connector B
flag =1
FALSE
flag = 0
FALSE
ENDDEFINE
TRUE
TRUE
WITH OPEN"user.txt" with WRITE mode as edit
DISPLAY ("THIS ACCOUNT IS NOT AVAILABLE")
edit_list
first recs
last recs
FALSE
DISPLAY "Successfully Withdrawal"
TRUE
WRITE ":" with frec.strip
DECLARE frec = ""
recs
FALSE
first field
last field
TRUE
frec += field +":"

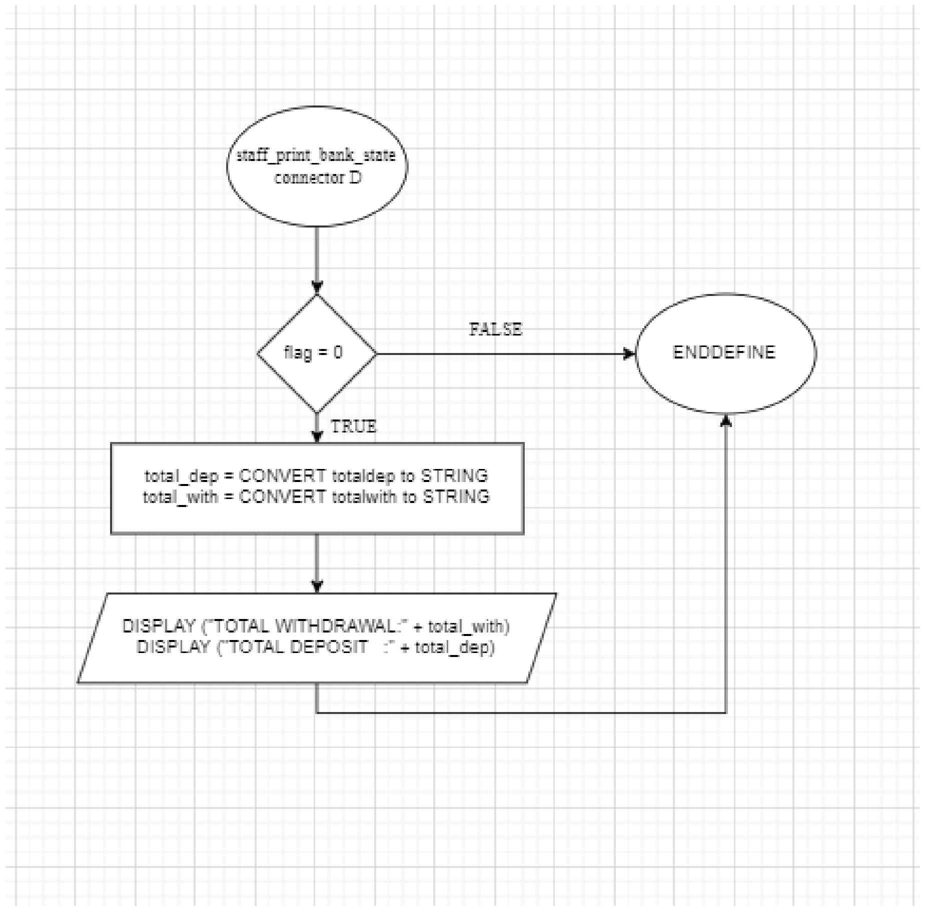
Staff print customer bank statement


DEFINE staff_print_bank_state()
DECLARE (STRING: year, years, month, months, day, details, detail, username, trans, tran, total_dep, total_with), (INTEGERS: totalwith,totaldep, balance), (LIST: rec, rect)
FROM datetime IMPORT datetime
now = datetime.datetime.now()
totalwith= 0
totaldep = 0
year = CONVERT now.year to STRING
years = CONVERT (now.year - 1) to STRING
month = CONVERT now.month to STRING
months = CONVERT (now.month - 1) to STRING
DISPLAY ("ACCOUNT NO") READ account
LENGTH of months=1
FALSE
TRUE
SET months = "0" + months
month = "03" OR month = "05" OR month = "07" OR month = "08" OR month = "10" OR month = "12
FALSE
month = "02" or month = "04" or month = "06" or month = "09" or month = "11"
TRUE
SET day ="31"
TRUE
SET day ="30"
FALSE
WITH OPEN "user.txt" with READ mode as detailread
details = detailread.readlines()
staff_print_bank_state connector A

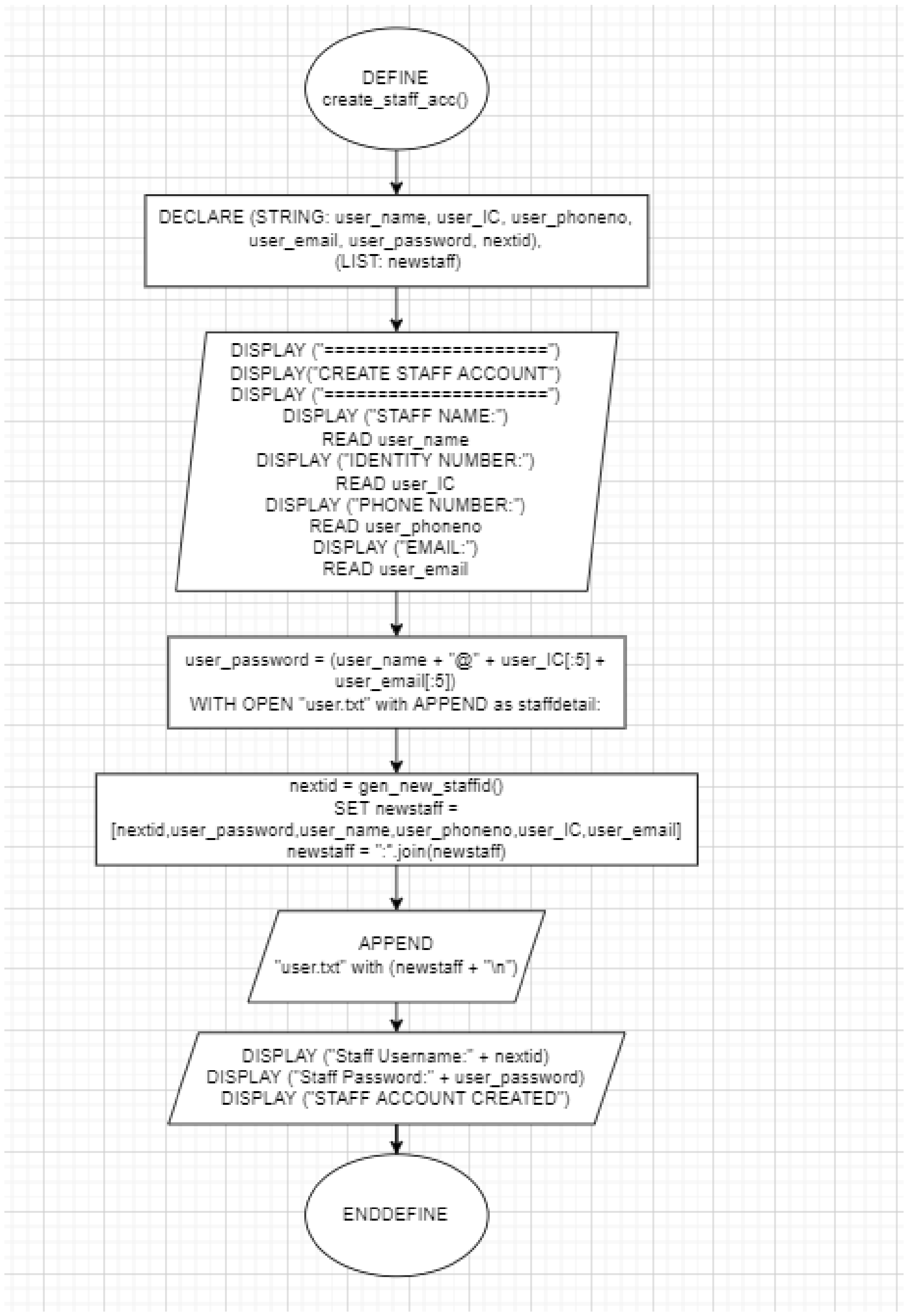

staff_print_bank_state connector A
details
first detail
last detail
FALSE
staff_print_bank_state connector B
TRUE
rec = detail. split(":")
account = rec [0]
FALSE
TRUE
BREAK
DISPLAY ("==============================")
DISPLAY ("BANK STATMENT")
DISPLAY("==============================")
DISPLAY (rec[4])
DISPLAY ("ACCOUNT NO:" + rec [0])
month = "01"
FALSE
TRUE
DISPLAY ("STATEMENT PERIOD:" + year + "/" + months + "/" + "01" + "-" + years + "/" + month + "/" + day)
DISPLAY ("STATEMENT PERIOD:" + years + "/" + "12" + "/" + "01" + "-" + year + "/" + month + "/" + "31")
DISPLAY ("--")
DISPLAY ("DATE".center(20),"DEPOSIT".center(20), "WITHDRAWAL".center(20),"BALANCE".center(20))
DISPLAY("--")


staff_print_bank_state connector B
account! =rec [0]
TRUE
DISPLAY ("THIS ACCOUNT IS NOT AVAILABLE")
SET flag = 1
FALSE
WITH OPEN "customerstatement.txt" with READ mode as transread:
trans = transread.readlines()
staff_print_bank_state connector C

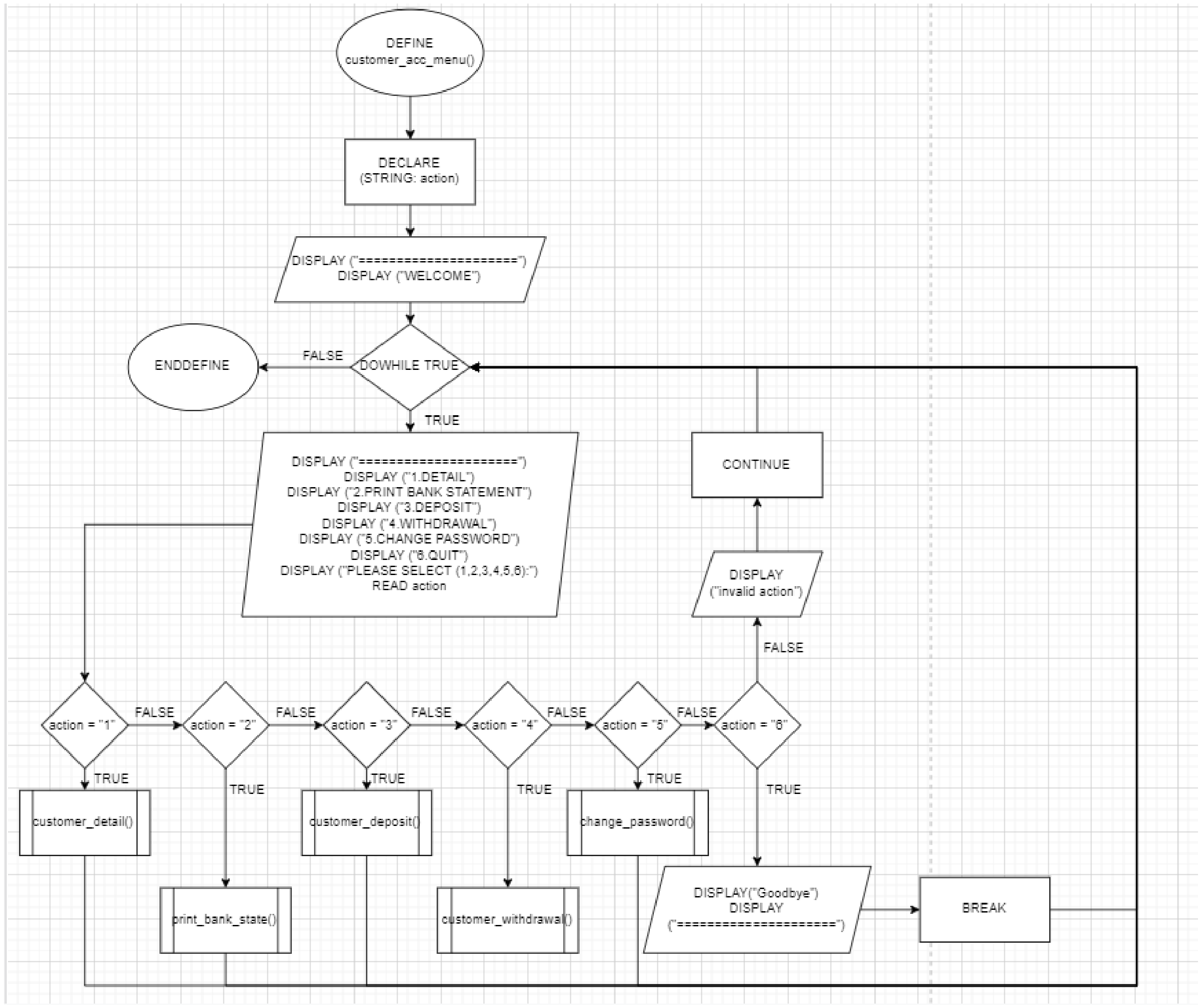

staff_print_bank_state connector C
trans
first tran
last tran
FALSE
staff_print_bank_state connector D
TRUE
rect = tran.split(":")
rect[2] = account
FALSE
TRUE
month = "01"
FALSE
TRUE
(rect[0])[:7] ==(year + "-" + month) or (rect[0])[:7] == (years + "-" + "12")
FALSE
(rect[0])[:7] =(year+ "-" + month) or (rect[0])[:7] = (year + "-" + months)
FALSE
TRUE
balance = CONVERT rect[3] to INTEGER
TRUE
balance = CONVERT rect[3] to INTEGER
rect[1] == "DEPOSIT"
FALSE
rect[1] == "WITHDRAWAL"
FALSE
TRUE
DISPLAY (rect[0].center(20),"".center(20), rect[3].center(20),rect[4].center(20))
totalwith = totalwith + balance
TRUE
DISPLAY (rect[0].center(20),rect[3].center(20), "".center(20),rect[4].center(20))
totaldep = totaldep + balance
rect[1] == "DEPOSIT"
FALSE
TRUE
DISPLAY (rect[0].center(20),rect[3].center(20), "".center(20),rect[4].center(20))
totaldep = totaldep + balance
rect[1] == "WITHDRAWAL"
FALSE
TRUE
DISPLAY (rect[0].center(20),"".center(20), rect[3].center(20),rect[4].center(20))
totalwith = totalwith + balance


staff_print_bank_state connector D
flag = 0
FALSE
TRUE
ENDDEFINE
total_dep = CONVERT totaldep to STRING
total_with = CONVERT totalwith to STRING
DISPLAY ("TOTAL WITHDRAWAL:" + total_with)
DISPLAY ("TOTAL DEPOSIT   :" + total_dep)

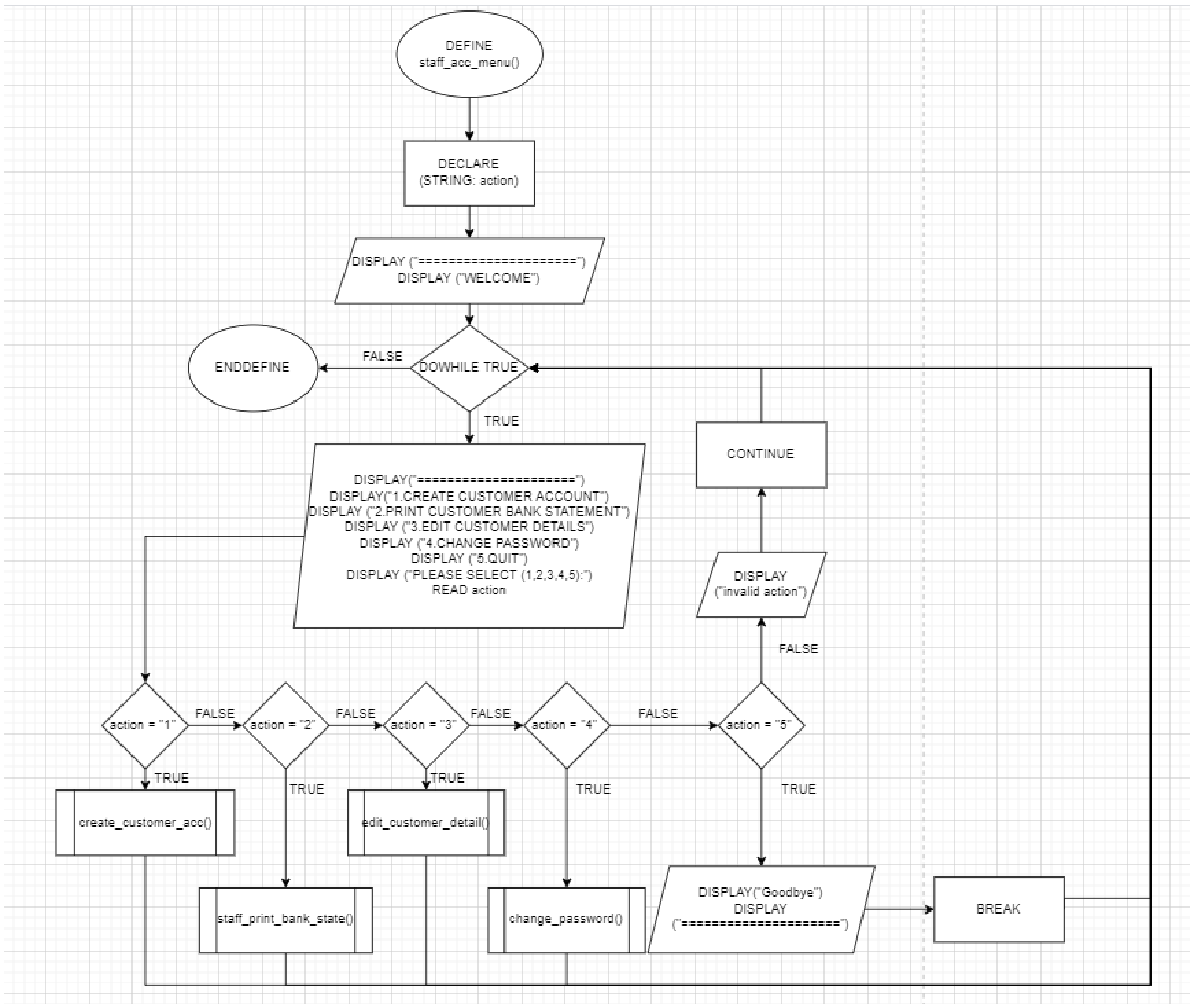
Super user creates staff account


DEFINE
create_staff_acc()
DECLARE (STRING: user_name, user_IC, user_phoneno, user_email, user_password, nextid), (LIST: newstaff)
DISPLAY ("=====================")
DISPLAY("CREATE STAFF ACCOUNT")
DISPLAY ("=====================")
DISPLAY ("STAFF NAME:")
READ user_name
DISPLAY ("IDENTITY NUMBER:")
READ user_IC
DISPLAY ("PHONE NUMBER:")
READ user_phoneno
DISPLAY ("EMAIL:")
READ user_email
user_password = (user_name + "@" + user_IC[:5] + user_email[:5])
WITH OPEN "user.txt" with APPEND as staffdetail:
nextid = gen_new_staffid()
SET newstaff = [nextid,user_password,user_name,user_phoneno,user_IC,user_email]
newstaff = ":".join(newstaff)
APPEND "user.txt" with (newstaff + "\n")
DISPLAY ("Staff Username:" + nextid)
DISPLAY ("Staff Password:" + user_password)
DISPLAY ("STAFF ACCOUNT CREATED")
ENDDEFINE

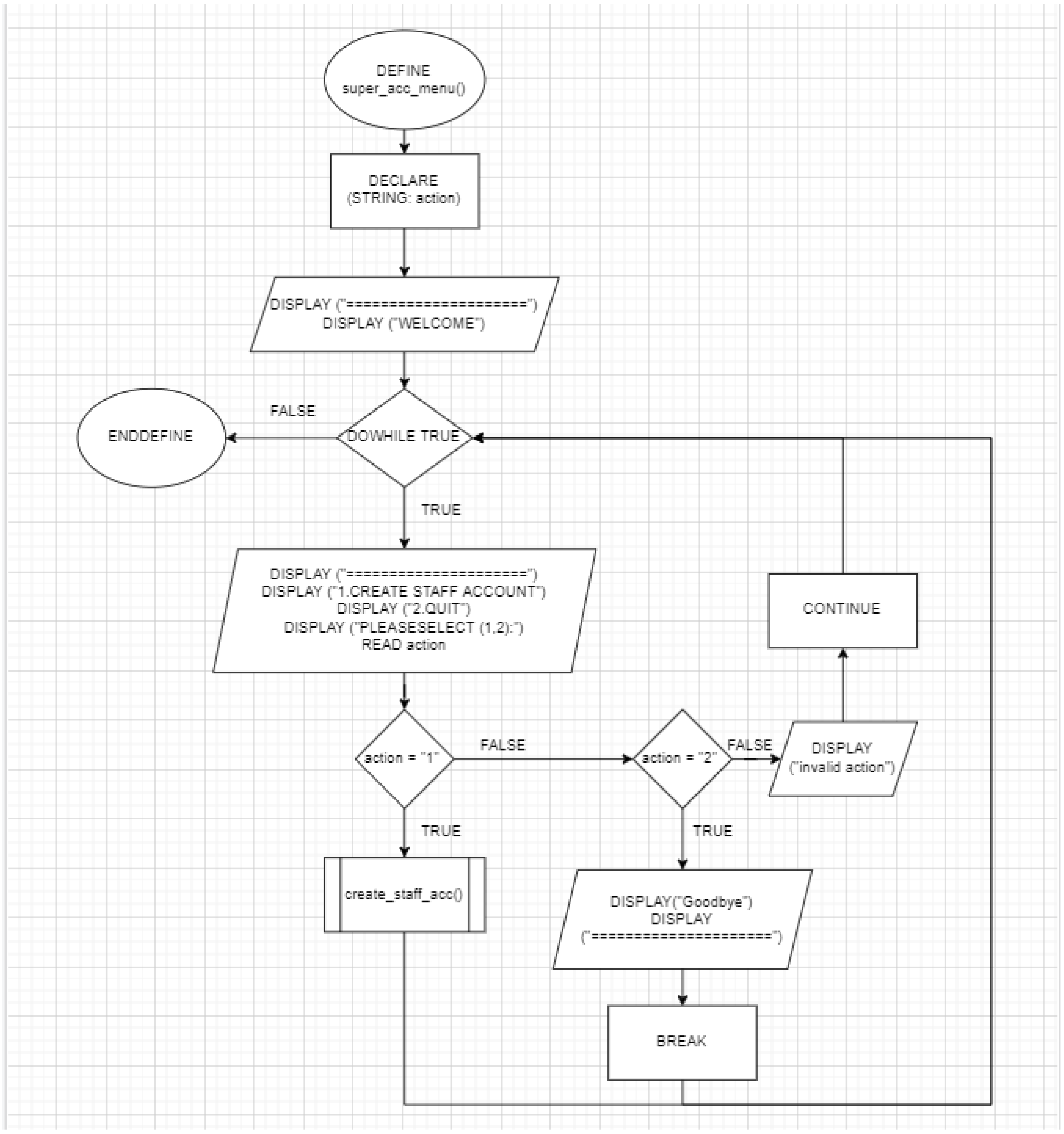
Customer account menu



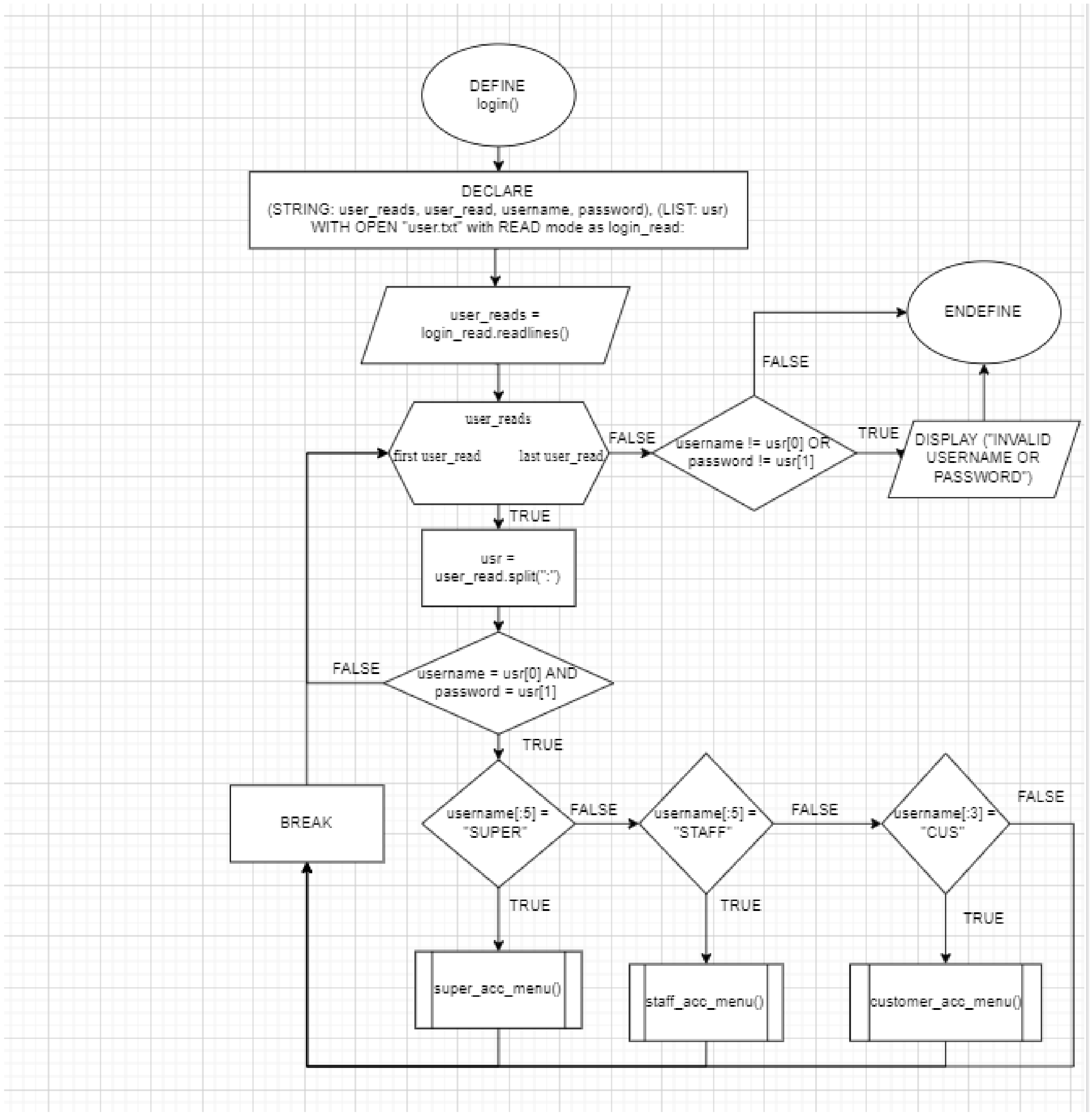
Staff account menu


DEFINE staff_acc_menu()
DECLARE (STRING: action)
DISPLAY ("====================") DISPLAY ("WELCOME")
DOWHILE TRUE
FALSE
ENDDEFINE
TRUE
DISPLAY("====================")
DISPLAY("1.CREATE CUSTOMER ACCOUNT")
DISPLAY ("2.PRINT CUSTOMER BANK STATEMENT")
DISPLAY ("3.EDIT CUSTOMER DETAILS")
DISPLAY ("4.CHANGE PASSWORD")
DISPLAY ("5.QUIT")
DISPLAY ("PLEASE SELECT (1,2,3,4,5):")
READ action
action = "1"
action = "2"
action = "3"
action = "4"
action = "5"
FALSE
TRUE
create_customer_acc()
staff_print_bank_state()
edit_customer_detail()
change_password()
DISPLAY("Goodbye") DISPLAY ("====================")
BREAK
DISPLAY ("invalid action")
CONTINUE

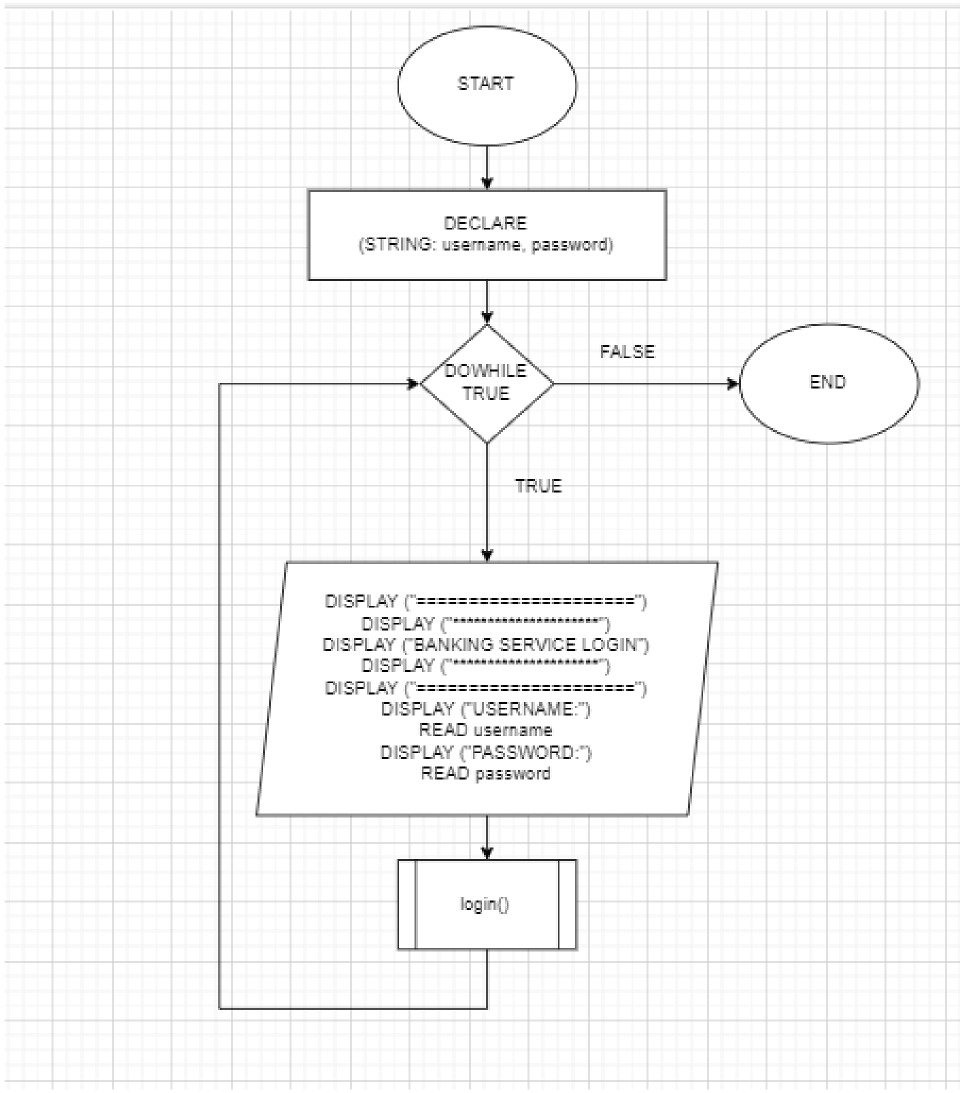
Super user account menu


DEFINE super_acc_menu()
DECLARE (STRING: action)
DISPLAY ("=====================") DISPLAY ("WELCOME")
DOWHILE TRUE
FALSE
ENDDEFINE
TRUE
DISPLAY ("=====================") DISPLAY ("1.CREATE STAFF ACCOUNT") DISPLAY ("2.QUIT") DISPLAY ("PLEASESELECT (1,2):") READ action
action = "1"
FALSE
action = "2"
FALSE
DISPLAY ("invalid action")
CONTINUE
TRUE
create_staff_acc()
TRUE
DISPLAY("Goodbye") DISPLAY ("=====================")
BREAK

User login


DEFINE
login()
DECLARE
(STRING: user_reads, user_read, username, password), (LIST: usr)
WITH OPEN "user.txt" with READ mode as login_read:
user_reads =
login_read.readlines()
user_reads
first user_read
last user_read
FALSE
username != usr[0] OR
password != usr[1]
TRUE
DISPLAY ("INVALID
USERNAME OR
PASSWORD")
FALSE
ENDEFINE
TRUE
usr =
user_read.split(":")
username = usr[0] AND
password = usr[1]
FALSE
TRUE
BREAK
username[:5] =
"SUPER"
FALSE
username[:5] =
"STAFF"
FALSE
username[:3] =
"CUS"
FALSE
TRUE
TRUE
TRUE
super_acc_menu()
staff_acc_menu()
customer_acc_menu()

Main logic


START
DECLARE
(STRING: username, password)
DOWHILE TRUE
FALSE
END
TRUE
DISPLAY ("=====================")
DISPLAY ("**********************")
DISPLAY ("BANKING SERVICE LOGIN")
DISPLAY ("**********************")
DISPLAY ("=====================")
DISPLAY ("USERNAME:")
READ username
DISPLAY ("PASSWORD:")
READ password
login()

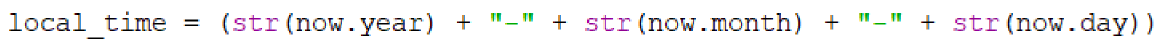
3.0 Programming concept with source code

In order to do coding using computer language such as python, c++, c#, java and etc. Source code is often used by programmers to learn about programming skill because source code is readable by human. However, source code is not a computer language and it cannot used to do coding. Variable, operator, control structure, repetition structure, exception handling, list, function and module are all example of source code.

3.1 Variable

In programming, programmers use variable to write data. Variable can store string, integer, list and etc. In a coding project a lot of variables are create to store different data. In order to teamwork with other in a programming project, a meaningful variable naming is very important. Space is not allowed for a variable name. So, a good programmer usually use underscore to replace space.

String

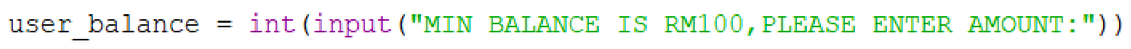


```
local_time = (str(now.year) + "-" + str(now.month) + "-" + str(now.day))
```

Figure 3.1.1

In figure 3.1.1, variable (local_time) are created and define by changing (now. year) to string. Numbers, letters, special characters are all consider as string.

Integer

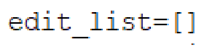


```
user_balance = int(input("MIN BALANCE IS RM100,PLEASE ENTER AMOUNT:"))
```

Figure 3.1.2

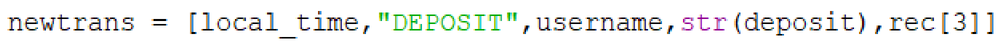
In figure 3.1.2, user can only input integer to variable (user_balance).

<u>List</u>



```
edit_list=[]
```

Figure 3.1.3



```
newtrans = [local_time,"DEPOSIT",username,str(deposit),rec[3]]
```

Figure 3.1.4

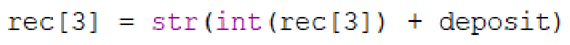
In figure 3.1.3, list (edit_list) are empty and string can add in by system. In figure 3.1.4 list (newtrans) contain different string inside.

3.2 Operator

Operator are useful when programmers want to do comparison for different condition or do calculation of variable.

<u>3.2.1 Arithmetic Operation</u>

<u>Addition</u>

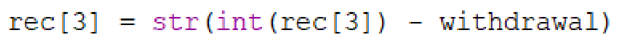


```
rec[3] = str(int(rec[3]) + deposit)
```

Figure 3.2.1

In figure 3.2.1, integer (rec [3]) are added with integer (deposit)

<u>Subtraction</u>

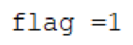


```
rec[3] = str(int(rec[3]) - withdrawal)
```

Figure 3.2.2

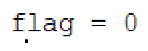
In figure 3.2.1, integer (withdrawal) are take out from integer (rec [3])

<u>3.2.2 Assignment operator</u>



```
flag =1
```

Figure 3.2.3



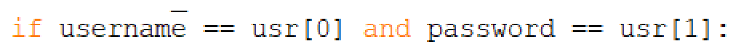
```
flag = 0
```

Figure 3.2.4

In figure 3.2.3, flag is initially set as integer 1. After different condition because of different input, flag has assigned to integer 0.

<u>3.3.3 comparison operator</u>

<u>Equal</u>

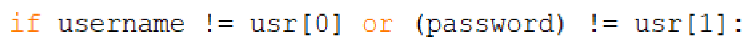


```
if username == usr[0] and password == usr[1]:
```

Figure 3.2.5

From figure 3.2.5, it shows that variable (username) and (password) are checking whether it is equal to usr [0] and usr [1].

<u>Not Equal</u>

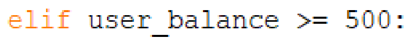


```
if username != usr[0] or (password) != usr[1]:
```

Figure 3.2.6

From figure 3.2.5, it shows that variable (username) and (password) are checking whether it is not equal to usr [0] and usr [1].

Greater or equal

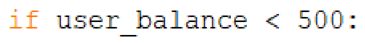


```
elif user_balance >= 500:
```

Figure 3.2.7

In figure 3.2.7 variable (user_balance) are checking whether it has greater or equal to 500.

Lesser



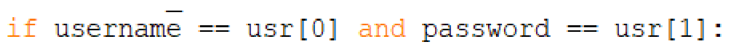
```
if user_balance < 500:
```

Figure 3.2.8

In figure 3.2.8 variable (user_balance) are checking whether it has lesser than 500.

3.3.4 Logical Operator

AND

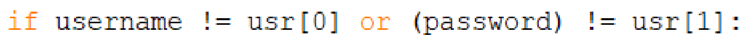


```
if username == usr[0] and password == usr[1]:
```

Figure 3.2.9

In figure 3.2.9, both comparison of variable (username == usr [0]) and (password == usr [1]) must be true or else one of them or both of them are false, the whole condition will become false.

OR



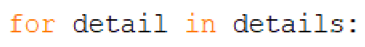
```
if username != usr[0] or (password) != usr[1]:
```

Figure 3.2.10

In figure 3.2.10, one of the statement or both of the statement are true then the whole condition is true unless both statements are false.

3.2.5 Membership Operator

IN



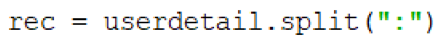
```
for detail in details:
```

Figure 3.2.11

In figure 3.2.11, the “detail” has been checked whether it is existed in “details”, if statement is true, it will do a for loop.

3.2.6 String Operator

Split

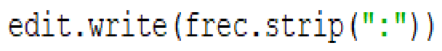


```
rec = userdetail.split(":")
```

Figure 3.2.12

In figure 3.2.12, rec has been defined as a list that contain string in userdetail separate by (“:”) using split.

Strip

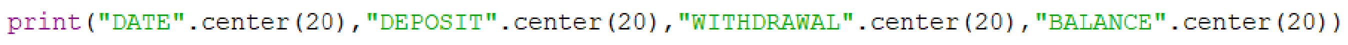


```
edit.write(frec.strip(":"))
```

Figure 3.2.12

Strip are used to remove specific character or unwanted space. In figure 3.2.12 strip has used to remove “:” at the last of the line.

String Center



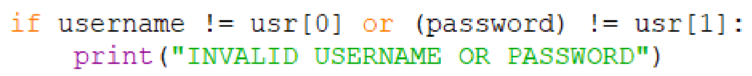
```
print("DATE".center(20),"DEPOSIT".center(20),"WITHDRAWAL".center(20),"BALANCE".center(20))
```

Figure 3.2.13

In figure 3.2.13, string has been centered by length of centered 20. The string will arrange accordingly. For example, “BALANCE” will center at the middle of length 60 – 80.

3.3 Control Structure

<u>If</u>

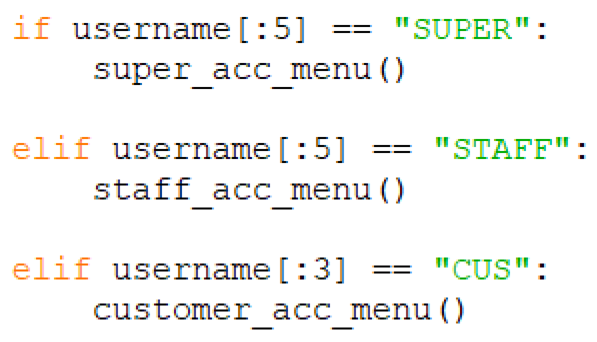


```
if username != usr[0] or (password) != usr[1]:
    print("INVALID USERNAME OR PASSWORD")
```

Figure 3.3.1

In figure 3.3.1, statement username! = usr[0] has been checked whether it is true of false. If it is true, the following instruction will process which is display the string.

<u>If-Elif</u>

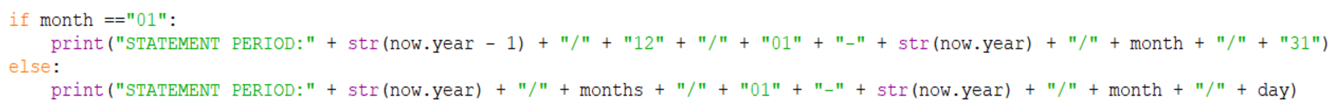


```
if username[:5] == "SUPER":
    super_acc_menu()

elif username[:5] == "STAFF":
    staff_acc_menu()

elif username[:3] == "CUS":
    customer_acc_menu()
```

Figure 3.3.2

In figure 3.3.2, the first statement username [:5] == “SUPER” has been checked. If it is true, function (super_acc_menu) will proceed. If it is false, the second statement username [:5] == “STAFF” will be checked true or false until the last statement.

<u>If-Else</u>

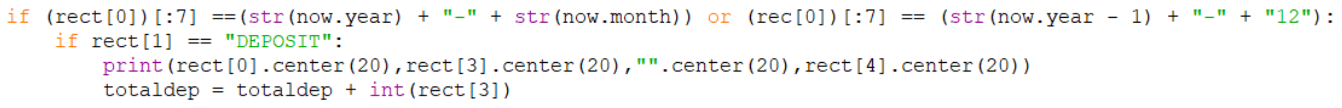


```
if month =="01":
    print("STATEMENT PERIOD:" + str(now.year - 1) + "/" + "12" + "/" + "01" + "-" + str(now.year) + "/" + month + "/" + "31")
else:
    print("STATEMENT PERIOD:" + str(now.year) + "/" + months + "/" + "01" + "-" + str(now.year) + "/" + month + "/" + day)
```

Figure 3.3.3

In figure 3.3.3, the statement month == “01” has been checked whether it is false or true. It will display different thing depends on statement is true or false.

Nested-If

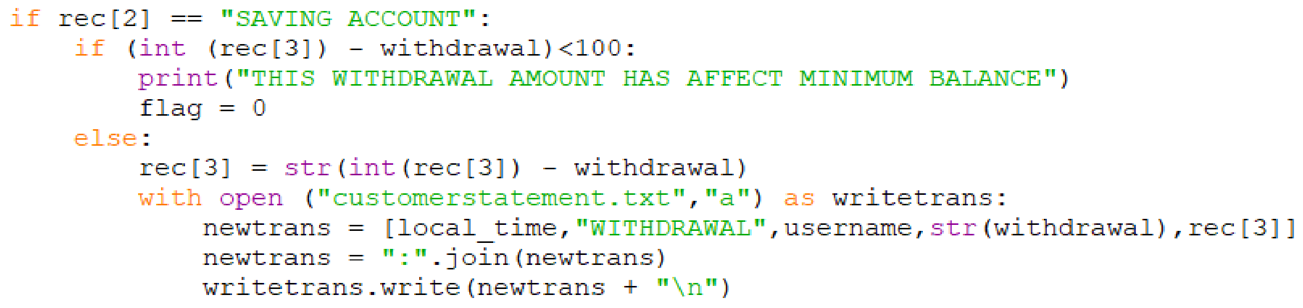


```
if (rect[0])[:7] ==(str(now.year) + "-" + str(now.month)) or (rec[0])[:7] == (str(now.year - 1) + "-" + "12"):
    if rect[1] == "DEPOSIT":
        print(rect[0].center(20),rect[3].center(20),"".center(20),rect[4].center(20))
        totaldep = totaldep + int(rect[3])
```

Figure 3.3.4

In figure 3.3.4, if (rect [0]) [:7] equal to the following string is true then it will proceed to check rect [1] == "DEPOSIT" is true. It will display the following string and do calculation of totaldep.

Nested-If-Else



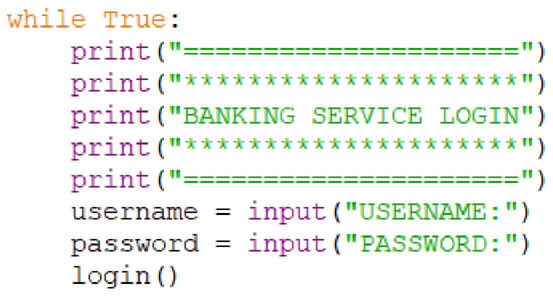
```
if rec[2] == "SAVING ACCOUNT":
    if (int (rec[3]) - withdrawal)<100:
        print("THIS WITHDRAWAL AMOUNT HAS AFFECT MINIMUM BALANCE")
        flag = 0
    else:
        rec[3] = str(int(rec[3]) - withdrawal)
        with open ("customerstatement.txt","a") as writetrans:
            newtrans = [local_time,"WITHDRAWAL",username,str(withdrawal),rec[3]]
            newtrans = ":".join(newtrans)
            writetrans.write(newtrans + "\n")
```

Figure 3.3.5

In figure 3.3.5, statement rec [2] == "SAVING ACCOUNT" will be check. If it is true, it will check the statement calculation lesser than 100.If it is true following instruction will proceed. If false another instruction will proceed.

3.4 Repetition Structure

While loop

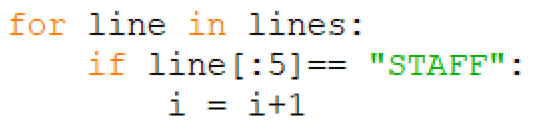


```
while True:
    print("=====================")
    print("*********************")
    print("BANKING SERVICE LOGIN")
    print("*********************")
    print("=====================")
    username = input("USERNAME:")
    password = input("PASSWORD:")
    login()
```

Figure 3.4.1

In figure 3.4.1, while True means loop forever. After those display, username password input and function login has proceeded finish it will back to the initial.

For loop

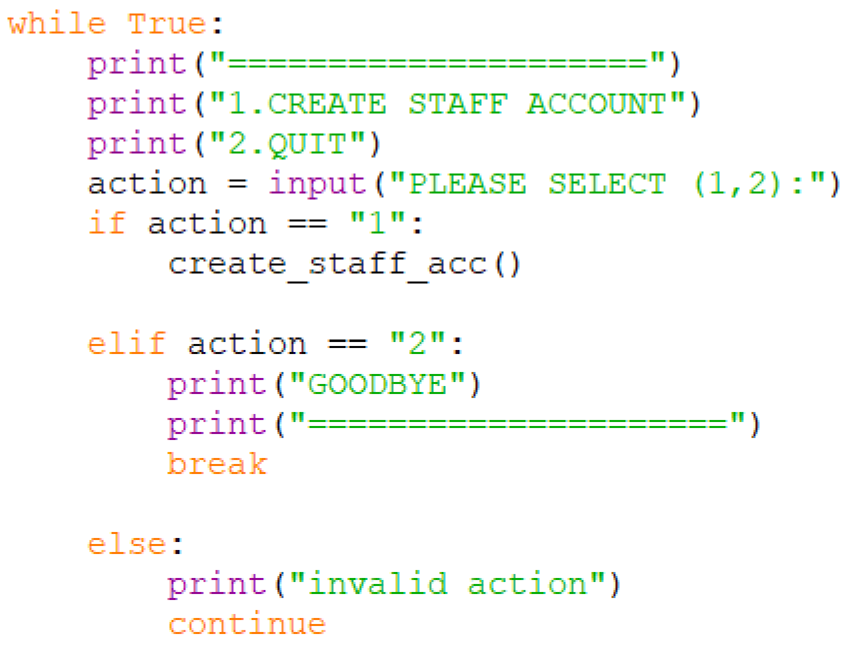


```
for line in lines:
    if line[:5]== "STAFF":
        i = i+1
```

Figure 3.4.2

In figure 3.4.2 line in lines is loop one by 1. If line [:5] == "STAFF" is true then integer i will add 1 until all line has been loop.

Break and Continue



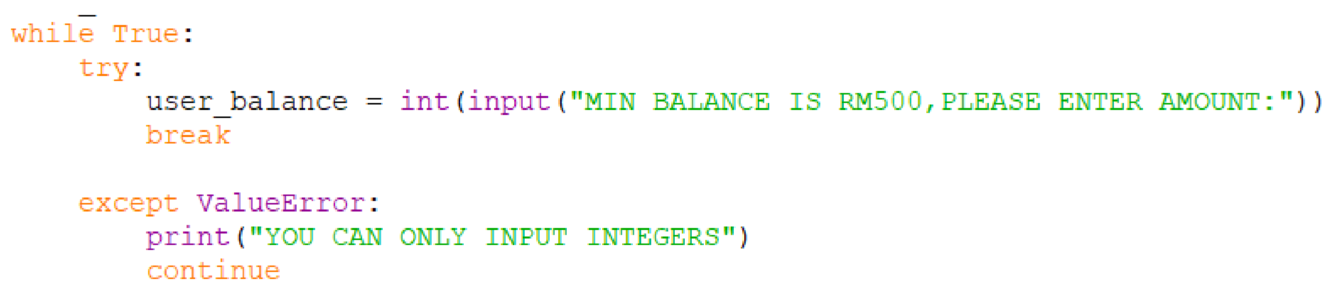
```
while True:
    print("=====================")
    print("1.CREATE STAFF ACCOUNT")
    print("2.QUIT")
    action = input("PLEASE SELECT (1,2):")
    if action == "1":
        create_staff_acc()

    elif action == "2":
        print("GOODBYE")
        print("=====================")
        break

    else:
        print("invalid action")
        continue
```

Figure 3.4.3

In figure 3.4.3, while looping, if action == "2" is true break will process to stop looping. If all statement is false, continue are used to back to initial and loop again.

3.5 Exception Handling

Try and Except



```
while True:
    try:
        user_balance = int(input("MIN BALANCE IS RM500,PLEASE ENTER AMOUNT:"))
        break

    except ValueError:
        print("YOU CAN ONLY INPUT INTEGERS")
        continue
```

Figure 3.5.1

In order to do validation and prevent system error. Try and except are useful to do that. In figure 3.5.1, if user_balance is input as integer then it will break out the loop. If it is not integer, it will display and continue the loop.

3.6 Function

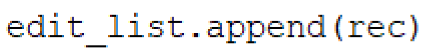
A good coding is not doing globally. In order to do locally, define are commonly used.

gen_new_cusid	This function is used to auto generate customer id.
gen_new_staffid	This function is used to auto generate staff id.
change_password	This function is used to modify password of staff and customer.
customer_detail	Customer can view their own detail by this function.
customer_deposit	Customer can use this function to deposit. Value will add and stored.

customer_withdrawal	Customer can use this function to withdraw. Value will take out and stored.
print_bank_state	This function can let customer see their bank statement.
create_customer_acc	Staff can use this function to create customer account.
edit_customer_detail	This function provides modify customer detail by staff.
staff_print_bank_state	Staff can print customer bank statement by input customer ID.
create_staff_acc	Default account can create staff account using this function.
customer_acc_menu	This function show customer account menu.
staff_acc_menu	This function show staffs account menu.
super_acc_menu	This function show default account menu.
login	This function verifies different account and check exist account.

3.7 List

Append



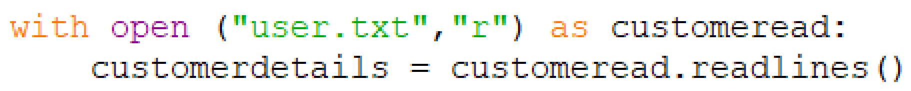
```
edit_list.append(rec)
```

Figure 3.7.1

In figure 3.7.1, rec have added to list by using append.

3.8 File

Read

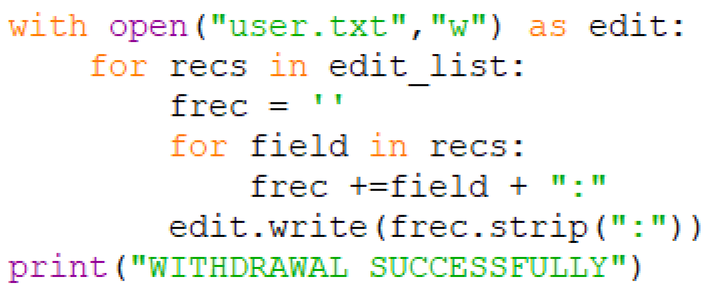


```
with open ("user.txt","r") as customeread:
    customerdetails = customeread.readlines()
```

Figure 3.8.1

In figure 3.8.1, “user.txt” has been open as read mode using variable customerread. Variable customerdetails are define as read lines in “user.txt”.

Write

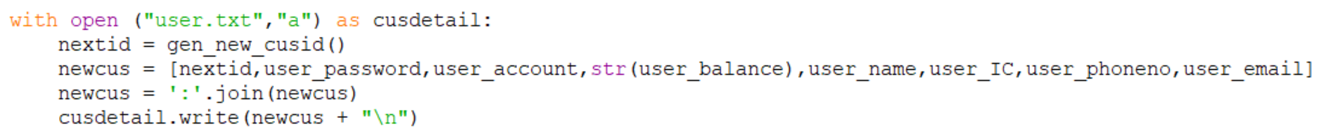


```
with open("user.txt","w") as edit:
    for recs in edit_list:
        frec = ''
        for field in recs:
            frec +=field + ":"
        edit.write(frec.strip(":"))
print("WITHDRAWAL SUCCESSFULLY")
```

Figure 3.8.2

In figure 3.8.2, “user.txt” has been open as write mode using variable edit. In the looping, frec has write into the file using write.

Append



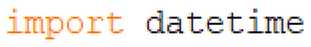
```
with open ("user.txt","a") as cusdetail:
    nextid = gen_new_cusid()
    newcus = [nextid,user_password,user_account,str(user_balance),user_name,user_IC,user_phoneno,user_email]
    newcus = ':'.join(newcus)
    cusdetail.write(newcus + "\n")
```

Figure 3.8.3

In figure 3.8.3, “user.txt” has been open as append mode using variable cusdetail. Variable newcus are added to “user.txt” using append.

3.9 Module

<u>**datetime**</u>



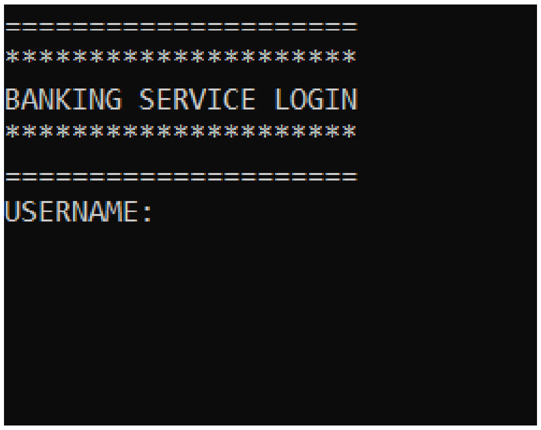
```
import datetime
```

Figure 3.9.1

Datetime is a module that provide function of making system having concept of date and time.

4.0 Sample of Output and Input

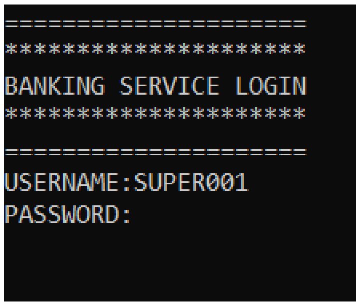
Login interface



```
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:
```

Figure 4.1

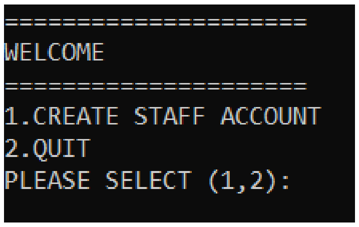
Figure 4.1 show the interface of login system.



```
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:SUPER001
PASSWORD:
```

Figure 4.2

Figure 4.2 show password required after input username.

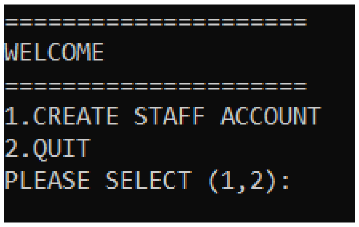


```
=====================
WELCOME
=====================
1.CREATE STAFF ACCOUNT
2.QUIT
PLEASE SELECT (1,2):
```

Figure 4.3

Figure 4.3 show login successfully to account depends on which type of account using.

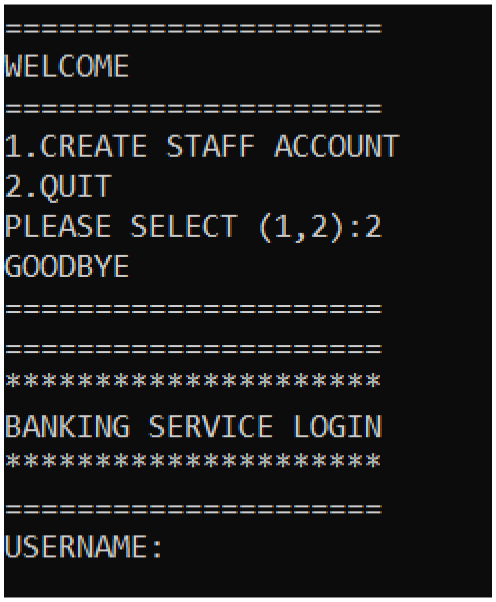
Super user - account menu



```
====================
WELCOME
====================
1.CREATE STAFF ACCOUNT
2.QUIT
PLEASE SELECT (1,2):
```

Figure 4.4

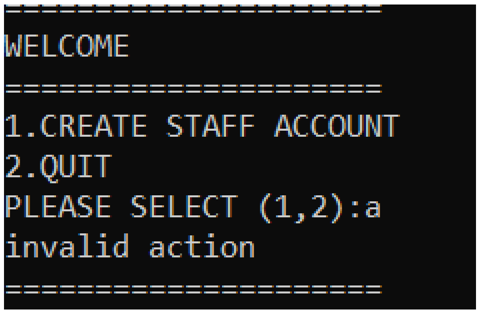
Figure 4.4 show the menu of super user account. Action are required to input.



```
=====================
WELCOME
=====================
1.CREATE STAFF ACCOUNT
2.QUIT
PLEASE SELECT (1,2):2
GOODBYE
=====================
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:
```

Figure 4.5

Figure 4.5 show action 2 can used to quit to the login interface.

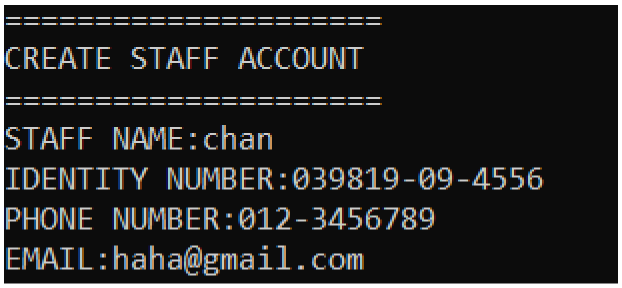


```
=====================
WELCOME
=====================
1.CREATE STAFF ACCOUNT
2.QUIT
PLEASE SELECT (1,2):a
invalid action
=====================
```

Figure 4.6

Figure 4.6 show only “1” and “2” can be input or else it will display invalid action.

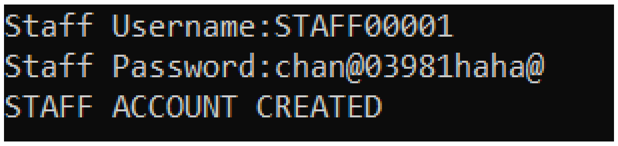
Super user - create staff account



```
=====================
CREATE STAFF ACCOUNT
=====================
STAFF NAME:chan
IDENTITY NUMBER:039819-09-4556
PHONE NUMBER:012-3456789
EMAIL:haha@gmail.com
```

Figure 4.6

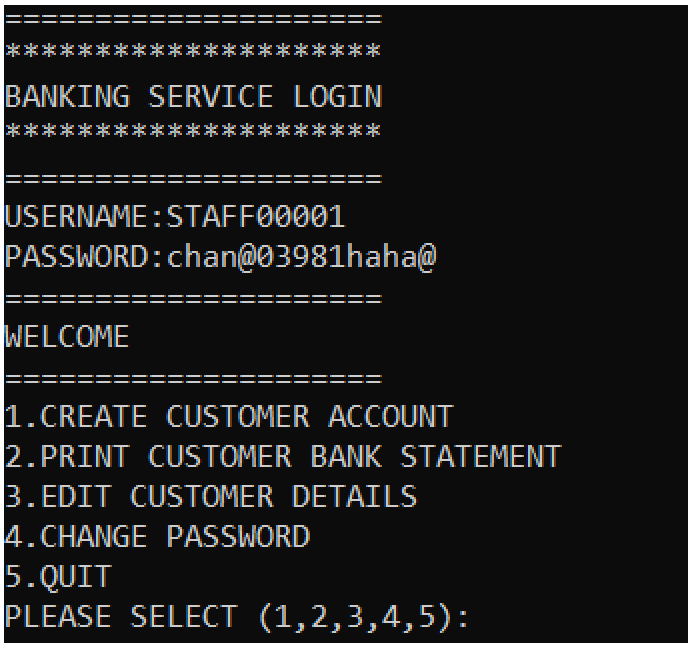
Figure 4.6 show detail of staff must be entered to system before create account.



```
Staff Username:STAFF00001
Staff Password:chan@03981haha@
STAFF ACCOUNT CREATED
```

Figure 4.7

Figure 4.7 show auto generate staff id and password have given and account are successfully created.

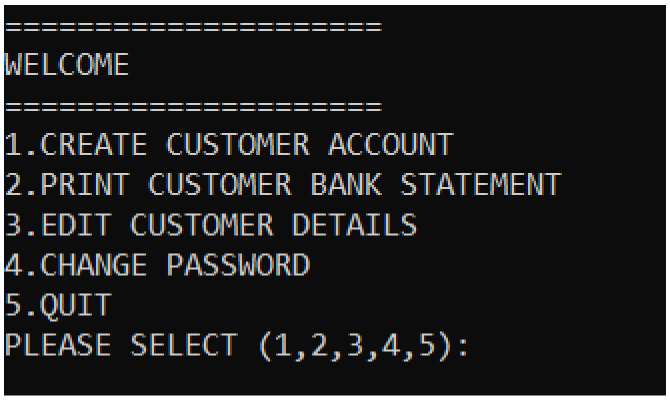


```
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:STAFF00001
PASSWORD:chan@03981haha@
=====================
WELCOME
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):
```

Figure 4.8

Figure 4.8 show we can login to staff account by entering given id and password.

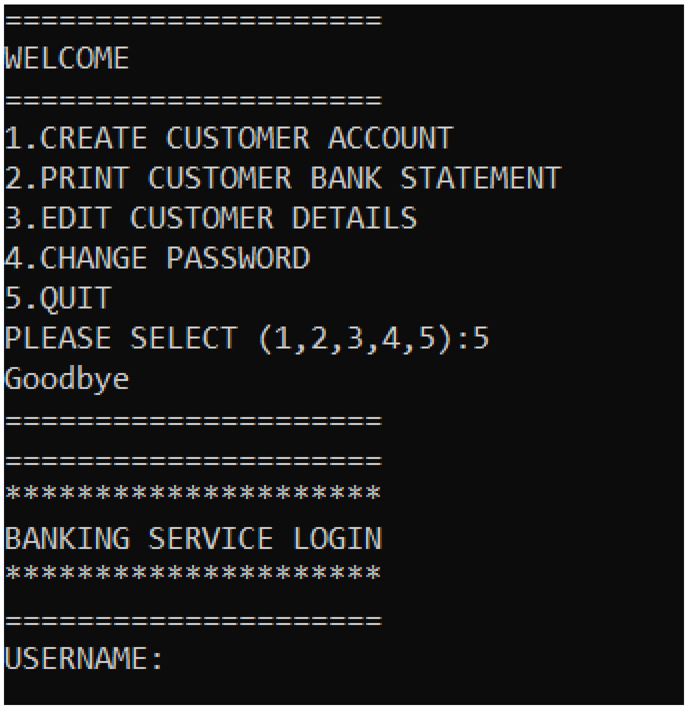
Staff user – account menu



```
=====================
WELCOME
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):
```

Figure 4.9

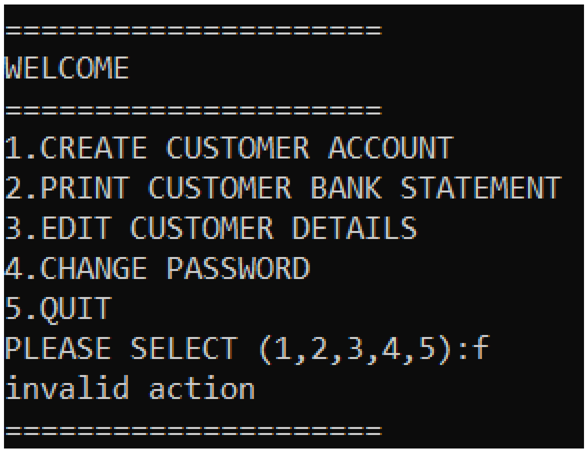
Figure 4.9 show the menu of staff account. Action is required to input by entering following number to do following function.



```
=====================
WELCOME
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):5
Goodbye
=====================
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:
```

Figure 4.10

Figure 4.10 show action “5” can use to quit and back to login interface.

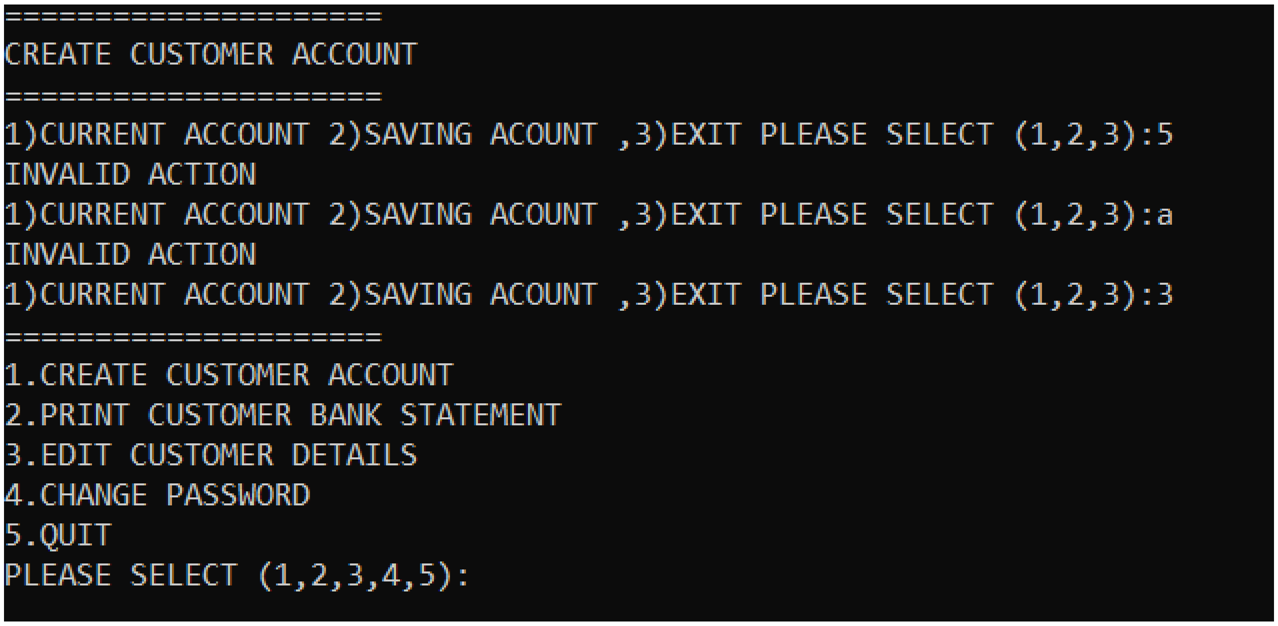


```
=====================
WELCOME
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):f
invalid action
=====================
```

Figure 4.11

Figure 4.11 show action except for 1,2,3,4,5 will display invalid action.

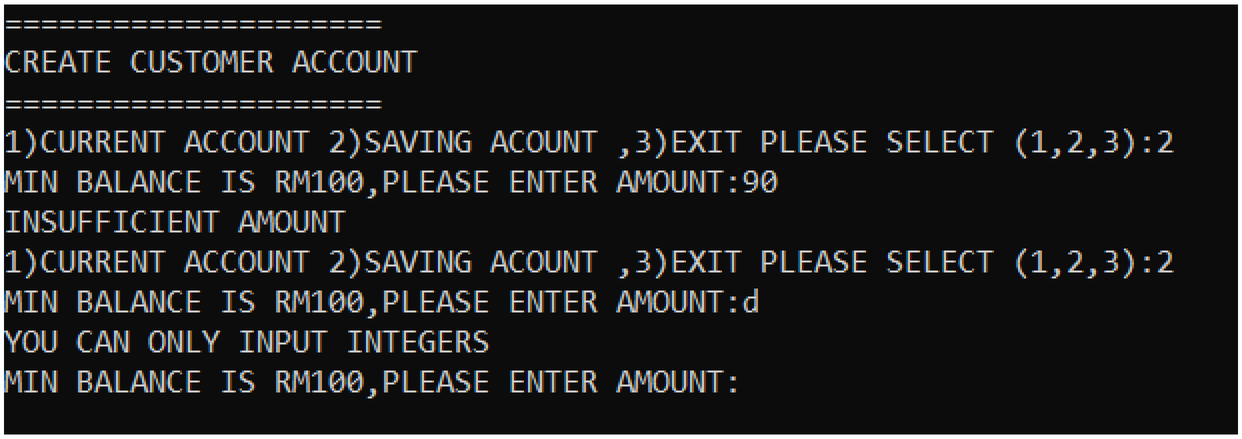
Staff user - create customer account



```
=====================
CREATE CUSTOMER ACCOUNT
=====================
1)CURRENT ACCOUNT 2)SAVING ACOUNT ,3)EXIT PLEASE SELECT (1,2,3):5
INVALID ACTION
1)CURRENT ACCOUNT 2)SAVING ACOUNT ,3)EXIT PLEASE SELECT (1,2,3):a
INVALID ACTION
1)CURRENT ACCOUNT 2)SAVING ACOUNT ,3)EXIT PLEASE SELECT (1,2,3):3
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):
```

Figure 4.12

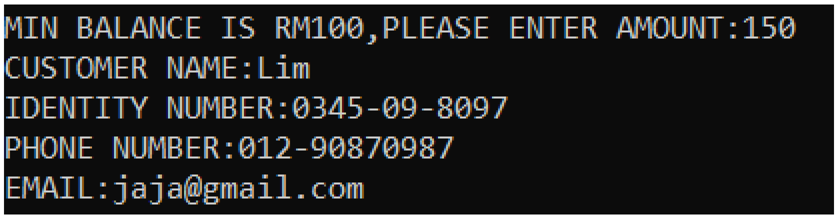
Figure 4.12 shows that we need to choose current account or saving account. Action except for 1,2,3 will display invalid action. Action 3 can use to quit and back to staff menu.



```
=====================
CREATE CUSTOMER ACCOUNT
=====================
1)CURRENT ACCOUNT 2)SAVING ACOUNT ,3)EXIT PLEASE SELECT (1,2,3):2
MIN BALANCE IS RM100,PLEASE ENTER AMOUNT:90
INSUFFICIENT AMOUNT
1)CURRENT ACCOUNT 2)SAVING ACOUNT ,3)EXIT PLEASE SELECT (1,2,3):2
MIN BALANCE IS RM100,PLEASE ENTER AMOUNT:d
YOU CAN ONLY INPUT INTEGERS
MIN BALANCE IS RM100,PLEASE ENTER AMOUNT:
```

Figure 4.13

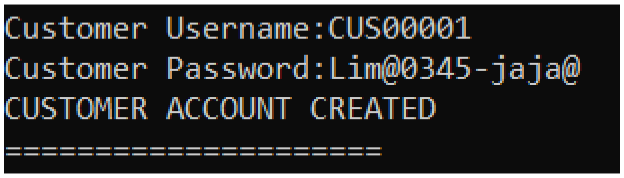
Figure 4.13 shows that insufficient amounts and not integer are input is not allowed to create customer account.



```
MIN BALANCE IS RM100,PLEASE ENTER AMOUNT:150
CUSTOMER NAME:Lim
IDENTITY NUMBER:0345-09-8097
PHONE NUMBER:012-90870987
EMAIL:jaja@gmail.com
```

Figure 4.14

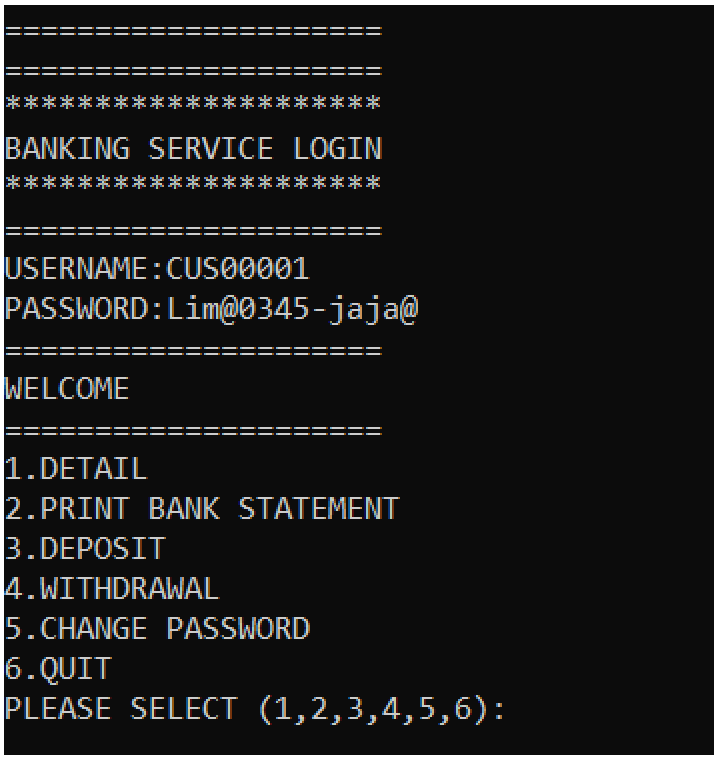
Figure 4.14 shows that details are required to create account.



```
Customer Username:CUS00001
Customer Password:Lim@0345-jaja@
CUSTOMER ACCOUNT CREATED
=====================
```

Figure 4.15

Figure 4.15 shows that customer id and password are given and account is successfully created.

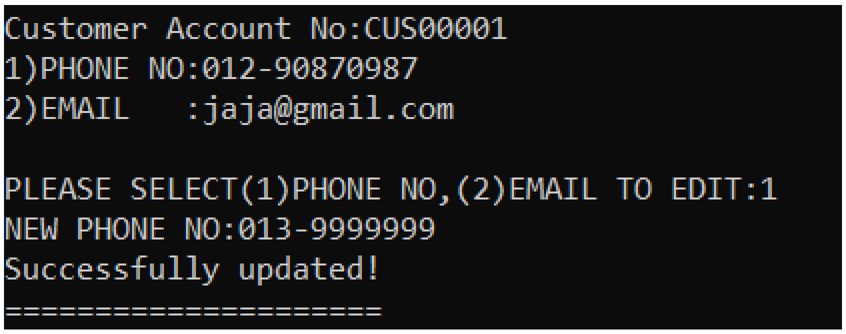


```
=====================
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:CUS00001
PASSWORD:Lim@0345-jaja@
=====================
WELCOME
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):
```

Figure 4.16

Figure 4.16 shows login to customer account is success.

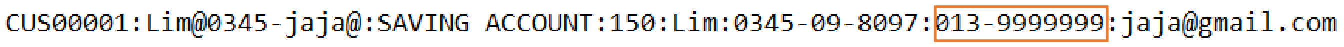
Staff user - edit customer details



```
Customer Account No:CUS00001
1)PHONE NO:012-90870987
2)EMAIL   :jaja@gmail.com

PLEASE SELECT(1)PHONE NO,(2)EMAIL TO EDIT:1
NEW PHONE NO:013-9999999
Successfully updated!
=====================
```

Figure 4.17

Figure 4.17 shows successfully edit customer details.

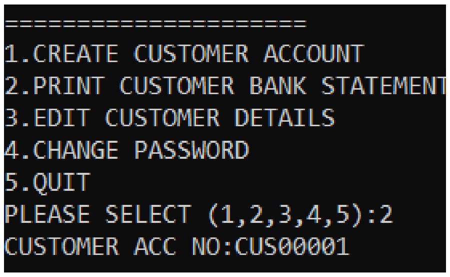


```
CUS00001:Lim@0345-jaja@:SAVING ACCOUNT:150:Lim:0345-09-8097:013-9999999:jaja@gmail.com
```

Figure 4.18

Figure 4.18 show details has been edited in txt file.

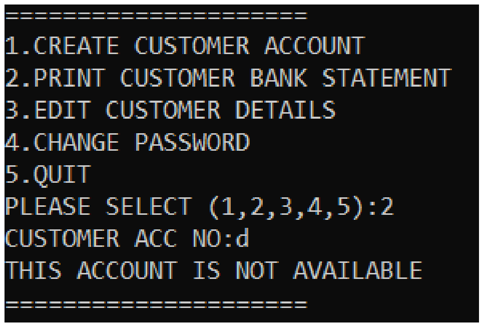
Staff user - print customer bank statement



```
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):2
CUSTOMER ACC NO:CUS00001
```

Figure 4.19

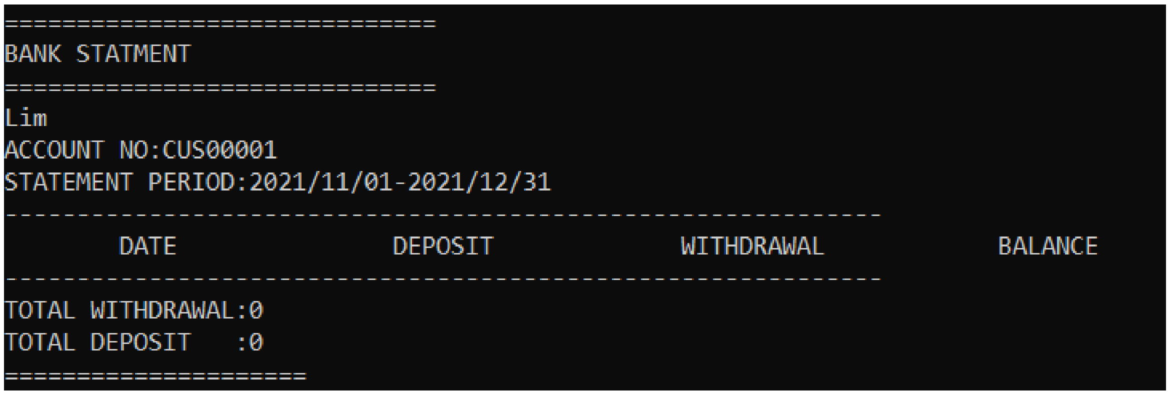
Figure 4.19 shows customer id are required to get bank statement.



```
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):2
CUSTOMER ACC NO:d
THIS ACCOUNT IS NOT AVAILABLE
=====================
```

Figure 4.20

Figure 4.20 shows that not existed account will display account is not available.

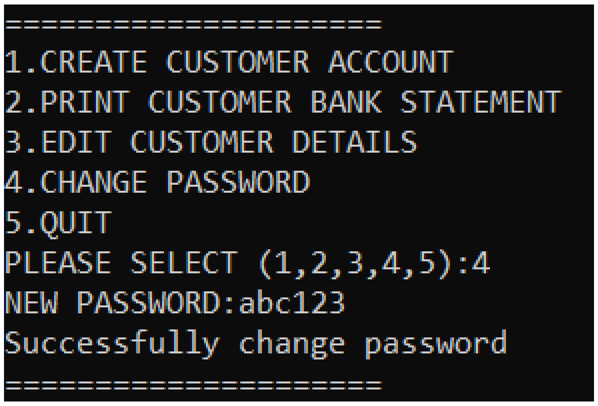


```
==============================
BANK STATMENT
==============================
Lim
ACCOUNT NO:CUS00001
STATEMENT PERIOD:2021/11/01-2021/12/31
-------------------------------------------------------------
        DATE                 DEPOSIT             WITHDRAWAL           BALANCE
-------------------------------------------------------------
TOTAL WITHDRAWAL:0
TOTAL DEPOSIT   :0
=====================
```

Figure 4.21

Figure 4.21 shows the bank statement of customer.

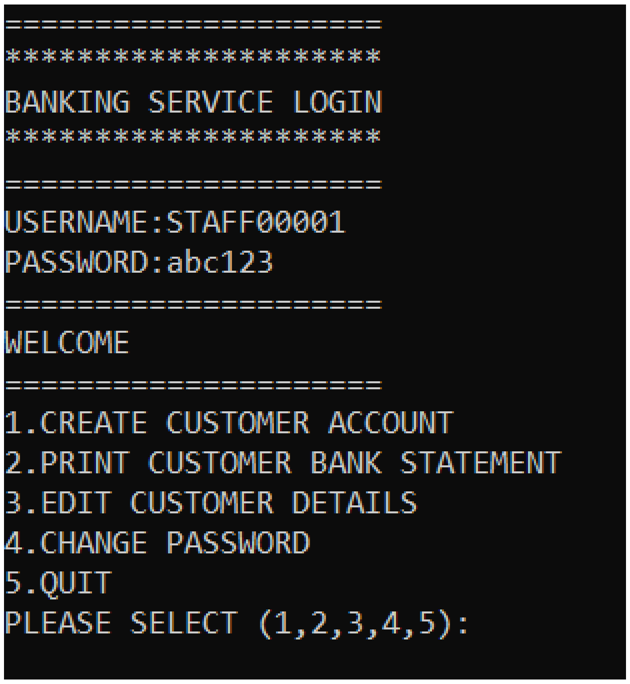
Staff user - change password



```
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):4
NEW PASSWORD:abc123
Successfully change password
=====================
```

Figure 4.22

Figure 4.22 shows password change successfully by entering new password.

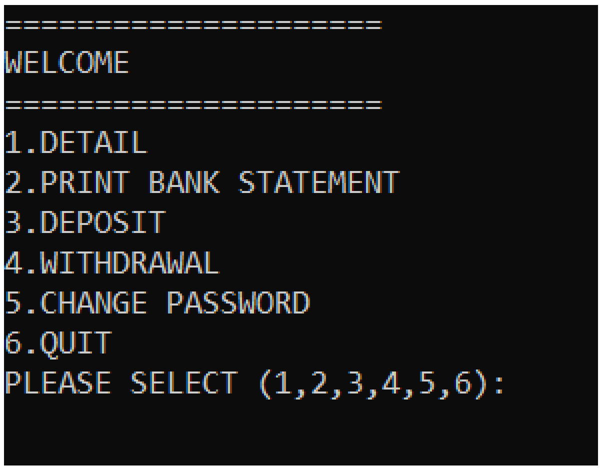


```
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:STAFF00001
PASSWORD:abc123
=====================
WELCOME
=====================
1.CREATE CUSTOMER ACCOUNT
2.PRINT CUSTOMER BANK STATEMENT
3.EDIT CUSTOMER DETAILS
4.CHANGE PASSWORD
5.QUIT
PLEASE SELECT (1,2,3,4,5):
```

Figure 4.23

Figure 4.23 shows staff can login to account by using new password.

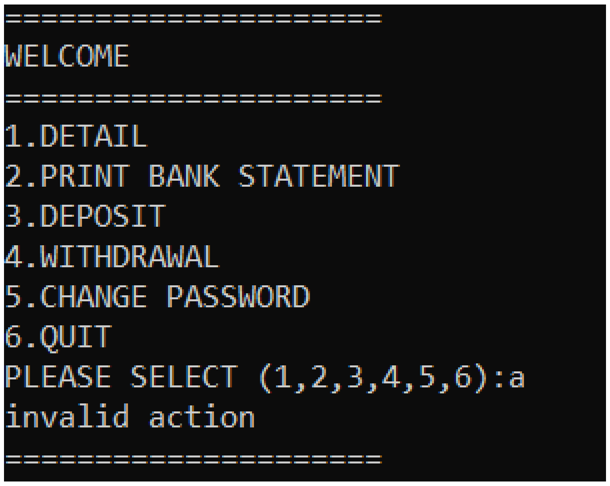
Customer user - account menu



```
=====================
WELCOME
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):
```

Figure 4.24

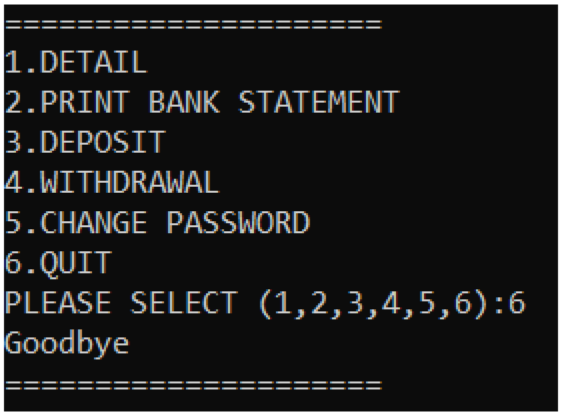
Figure 4.24 is the menu of customer account.



```
=====================
WELCOME
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):a
invalid action
=====================
```

Figure 4.25

Figure 4.25 shows that action other than 1,2,3,4,5,6 are invalid.

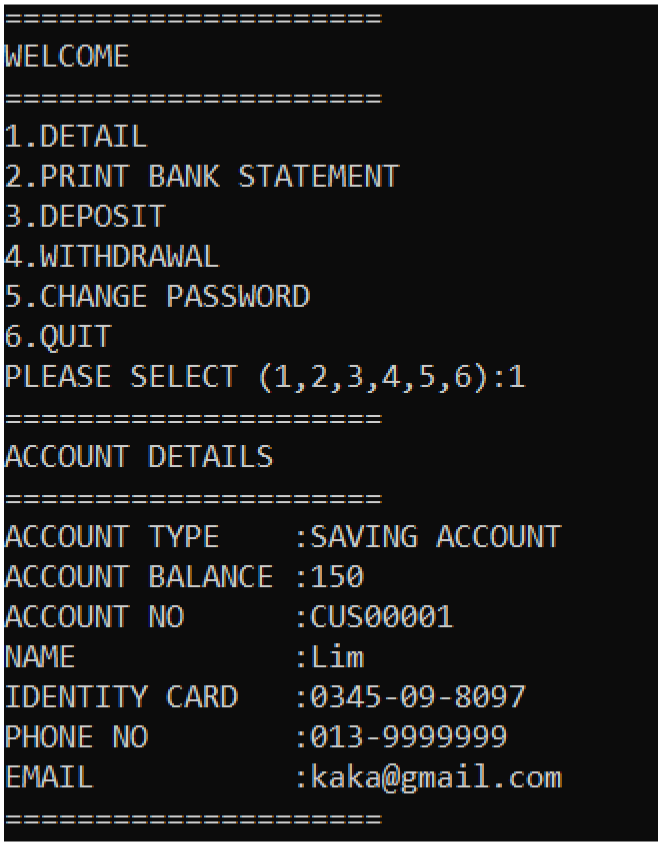


```
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):6
Goodbye
=====================
```

Figure 4.26

Figure 4.26 shows that action 6 will quit to login interface.

Customer user - view details

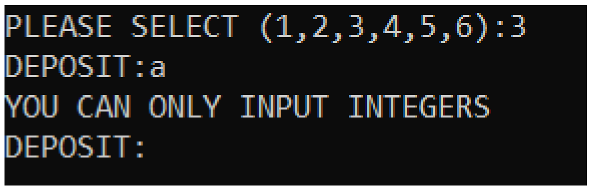


```
=====================
WELCOME
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):1
=====================
ACCOUNT DETAILS
=====================
ACCOUNT TYPE    :SAVING ACCOUNT
ACCOUNT BALANCE :150
ACCOUNT NO      :CUS00001
NAME            :Lim
IDENTITY CARD   :0345-09-8097
PHONE NO        :013-9999999
EMAIL           :kaka@gmail.com
=====================
```

Figure 4.27

Figure 4.27 shows the account details of customer.

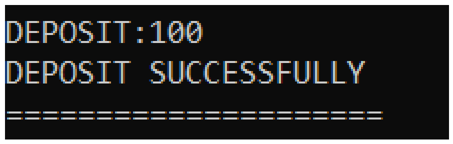
Customer user - deposit



```
PLEASE SELECT (1,2,3,4,5,6):3
DEPOSIT:a
YOU CAN ONLY INPUT INTEGERS
DEPOSIT:
```

Figure 4.28

Figure 4.28 input of amount are required to enter to system. Non -integers are not allowed to input or else it will loop again.

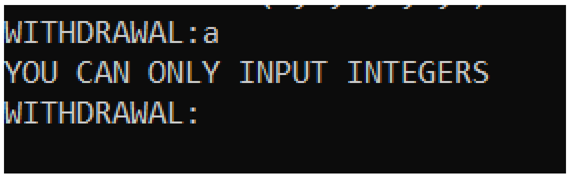


```
DEPOSIT:100
DEPOSIT SUCCESSFULLY
=====================
```

Figure 4.29

Figure 4.29 shows deposit successfully.

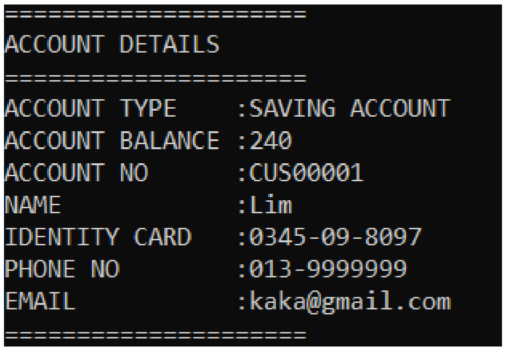
Customer user – withdrawal



```
WITHDRAWAL:a
YOU CAN ONLY INPUT INTEGERS
WITHDRAWAL:
```

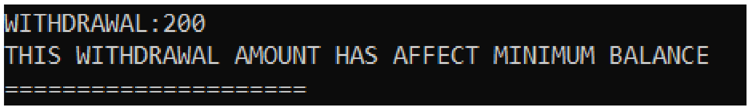
Figure 4.30

Figure 4.30 shows only integer can be input.



```
=====================
ACCOUNT DETAILS
=====================
ACCOUNT TYPE    :SAVING ACCOUNT
ACCOUNT BALANCE :240
ACCOUNT NO      :CUS00001
NAME            :Lim
IDENTITY CARD   :0345-09-8097
PHONE NO        :013-9999999
EMAIL           :kaka@gmail.com
=====================
```

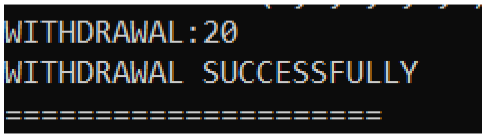
Figure 4.31



```
WITHDRAWAL:200
THIS WITHDRAWAL AMOUNT HAS AFFECT MINIMUM BALANCE
=====================
```

Figure 4.32

Figure 4.31 shows the account only have balance of 240. If the withdraw amount make the balance less than 100 the withdraw will unsuccessful like figure 4.32 shows.

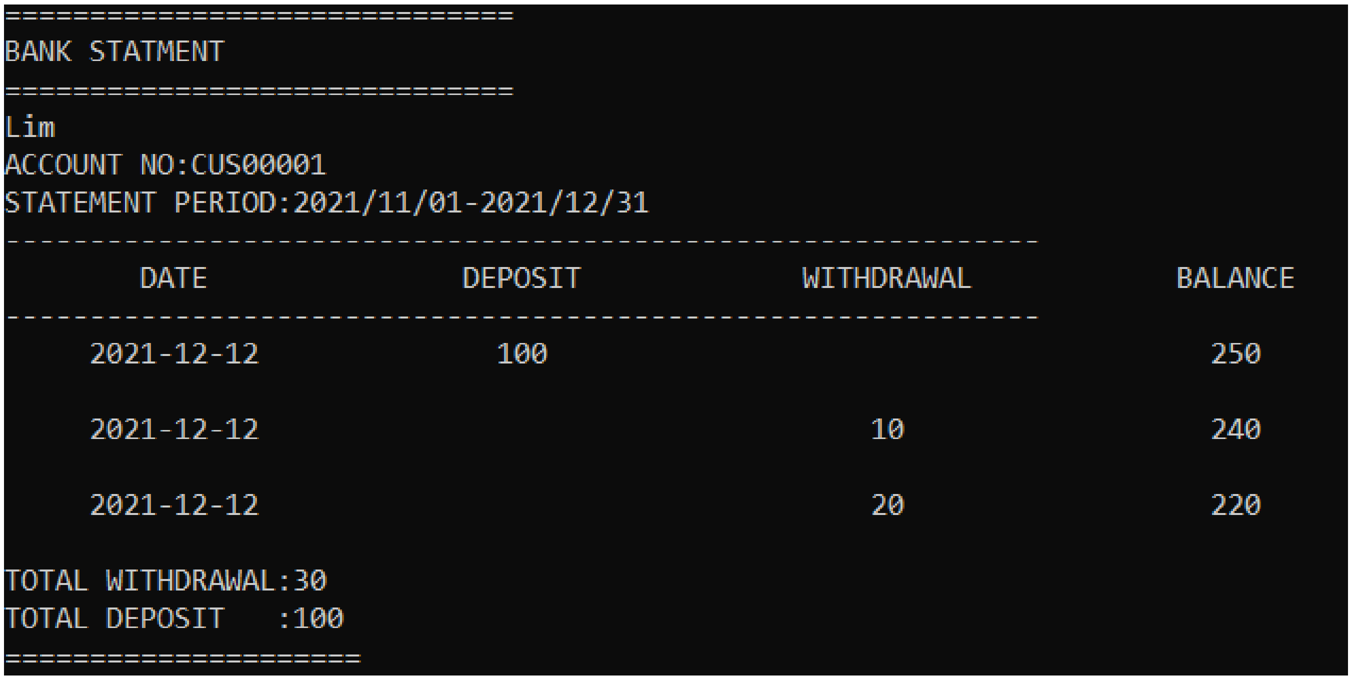


```
WITHDRAWAL:20
WITHDRAWAL SUCCESSFULLY
=====================
```

Figure 4.32

Figure 4.32 shows withdraw successfully.

Customer user - print bank statement

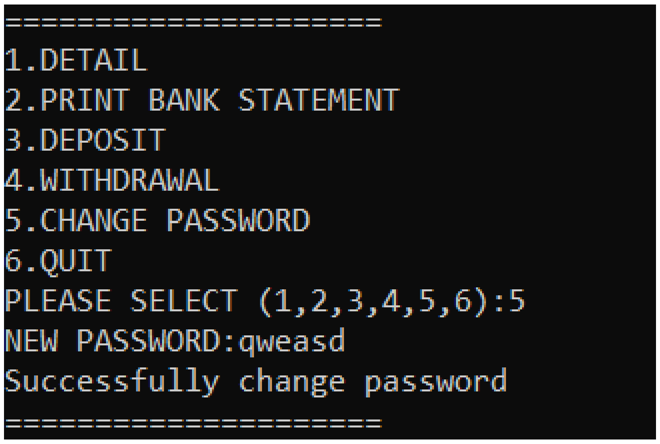


```
==============================
BANK STATMENT
==============================
Lim
ACCOUNT NO:CUS00001
STATEMENT PERIOD:2021/11/01-2021/12/31
-------------------------------------------------------------
        DATE                  DEPOSIT               WITHDRAWAL            BALANCE
-------------------------------------------------------------
     2021-12-12                 100                                         250

     2021-12-12                                         10                  240

     2021-12-12                                         20                  220

TOTAL WITHDRAWAL:30
TOTAL DEPOSIT   :100
=====================
```

Figure 4.33

Figure 4.33 shows bank statement of customer.

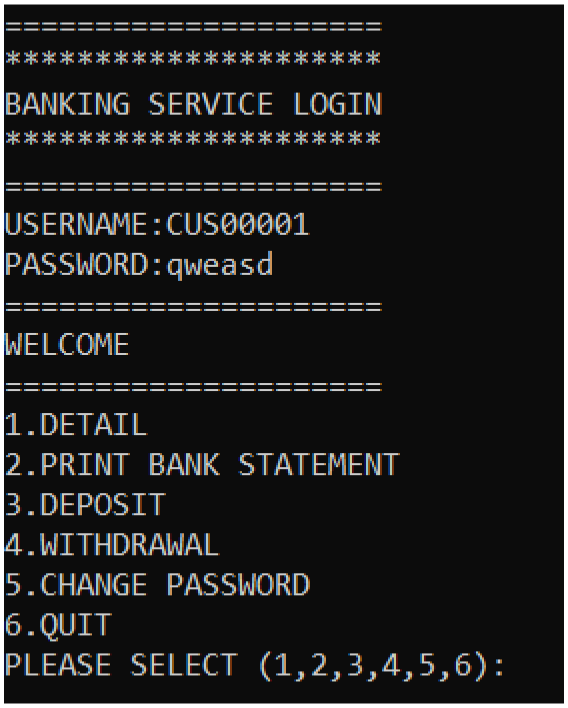
Customer user - change password



```
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):5
NEW PASSWORD:qweasd
Successfully change password
=====================
```

Figure 4.34

Figure 4.34 shows customer password has changed by input new password.



```
=====================
*********************
BANKING SERVICE LOGIN
*********************
=====================
USERNAME:CUS00001
PASSWORD:qweasd
=====================
WELCOME
=====================
1.DETAIL
2.PRINT BANK STATEMENT
3.DEPOSIT
4.WITHDRAWAL
5.CHANGE PASSWORD
6.QUIT
PLEASE SELECT (1,2,3,4,5,6):
```

Figure 4.35

Figure 4.35 shows customer can use new password to login to account.

5.0 Conclusion

As a conclusion, after one month of doing this assignment, I realized that this assignment has help me a lot. I have learned many functions inside python. I also know how to do validation of system and the important of validation. In order to make system efficiency, a good structure control is also important. By coding a banking system, I hope that I can help me gain more experience for future in work. From this assignment, I believe that I can step higher to become a good programmer.

6.0 Reference

1) The meaning of strip

https://careerkarma.com/blog/python-string-strip/

2)How to use split

https://www.w3schools.com/python/ref_string_split.asp

3)The information of bank

https://en.wikipedia.org/wiki/Bank

4)How to use read and write text file

https://www.youtube.com/watch?v=Uh2ebFW8OYM&t=476s

5)import of datetime

https://www.w3schools.com/python/python_datetime.asp