



# **INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT108-3-1-PYP**

**PYTHON PROGRAMMING**

**APDF1F2109PE, APU1F2109IT, APU1F2109MMT, APU1F2109PE,  
APU1F2109SE, APU1F2109TE, APU1F2109ME, APD1F2109IT,  
APD1F2109CE, APD1F2109EEE, APD1F2109ME, APU1F2109EEE,  
APU1F2109CE, APU1F2109CGD, APD1F2109CGD, APD1F2109TE,  
APD1F2109SE, APD1F2109MMT**

**HAND OUT DATE: 31<sup>st</sup> OCTOBER 2021**

**HAND IN DATE: 12<sup>th</sup> DECEMBER 2021**

**WEIGHTAGE: 100%**

---

## **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 Content**

### Contents

<b>Table of Content</b> .....	2
<b>Introduction</b> .....	3
<b>Assumptions</b> .....	3
<b>Design of the Program</b> .....	4
<b>Pseudocode</b> .....	4
<b>Flowchart</b> .....	23
<b>Programming Concepts with source code for explanation</b> .....	60
<b>Screenshots of samples and explanation</b> .....	74
<b>Conclusion</b> .....	97
<b>Reference</b> .....	98

## **Introduction**

This report is to demonstrate a system program named APU bank for banking service and handle all the primary information required to maintain the accounts of the customers in a bank. There are explanations, pseudocode and flowchart included for better understanding from different perspective. In this system, users including customers, super user and bank admin staff have their own account number, username and passwords. For customers, there are saving and current accounts for deposit and withdrawal purposes. The difference of saving accounts and current accounts is that they have different minimum balance. While for admin, they have the ability to create customer accounts, check and edit customer account details. Besides, printing customer's statement is also one of the functions admins can access so as super user. The super user's main function is to create admin accounts.

## **Assumptions**

1. I assume that all customers are Malaysians and have IC number.
2. I assume that only a limited number of staffs know how to login to super user.
3. I assume that all customers have a phone number from 8 digit to 10 digit.
4. I assume that customers name might have numbers.
5. I assume that admins will not forget their own username and passwords.
6. I assume that customers will not forget their own account number and passwords.
7. I assume that customers will change their password immediately after account is registered.
8. I assume that admin will only edit customer detail when asked to.
9. I assume that only admins are able to create customer account.
10. I assume that customers will not want to edit their name.
11. I assume that customers do not have the ability to edit their info themselves.
12. I assume that staff cannot login to customer's account.
13. I assume that customers cannot login to staff's account.

## **Design of the Program**

### **Pseudocode**

#### **Menu**

```
DEFINE menu
    DECLARE option AS INTEGER

    PRINT('-----Welcome to APU bank-----')
    PRINT("=====")
    PRINT ('Which account do you want to login? 1.Customer account 2.Admin')
    INPUT option
    IF option = 1
        CALL function logincustomer
    ELSEIF option = 2
        CALL function loginadmin
    ELSE
        PRINT ('Invalid input, please enter again!')
        CALL function menu
    ENDIF
```

### Customer's login

```
DEFINE logincustomer
  DECLARE accountnumber, customerpassword,
  DECLARE loginlist AS LIST
  DECLARE success AS BOOLEAN

  WHILE True
    PRINT('Please enter your account number:')
    INPUT accountnumber
    PRINT('Please enter your password:')
    INPUT customerpassword
    success = False
    OPEN 'Customerlogin.txt' AS customerlogin in READ MODE
    FOR i in customerlogin
      STRIP and SPLIT i USING ',' AS DELIMITER to loginlist
      IF (loginlist[0] EQUALS TO accountnumber and loginlist[1] EQUALS TO customerpassword)
        success = True
        BREAK LOOP
      ENDIF
    ENDFOR
    IF (success)
      PRINT('Login successful!')
      CALL function customermenu
    ELSE
      PRINT('Invalid account number and password!')
      PRINT('Please login again!')
      CALL function logincustomer
    ENDIF
  ENDWHILE
```

Customer's menu

```

DEFINE customermenu
    DECLARE customerchoice, customerchoice2 AS INTEGER

    PRINT('-----You are in customer menu-----')
    PRINT('=====')
    PRINT('Please choose an option 1.Withdraw 2.Deposit 3.Balance inquiry 4.Change passwrod 5.Exit')
    INPUT customerchoice
    IF customerchoice = 1
        WHILE True
            PRINT('Withdraw from 1.Saving 2.Current 3.Exit')
            INPUT customerchoice2
            IF customerchoice2 = 1
                CALL function savingwithdrawal
            ELSEIF customerchoice2 = 2
                CALL function currentwithdrawal
            ELSEIF customerchoice2 = 3
                CALL function customermenu
                BREAK LOOP
            ELSE
                PRINT('Please enter a valid input!')
            ENDIF
        ENDWHILE
    ELSEIF customerchoice = 2
        WHILE True
            PRINT('Deposit to 1.Saving 2.Current 3.Exit')
            INPUT customerchoice2
            IF customerchoice2 = 1
                CALL function savingdeposit
            ELSEIF customerchoice2 = 2
                CALL function currentdeposit
            ELSEIF customerchoice2 = 3
                CALL function customermenu
                BREAK LOOP
            ELSE
                PRINT('Please enter a valid input!')
            ENDIF
        ENDWHILE
    ELSEIF customerchoice = 3
        CALL function balanceinquiry
    ELSEIF customerchoice = 4
        CALL function changecustomerpassword
    ELSEIF customerchoice = 5
        PRINT('Thank you for using APU bank!')
        PRINT('Hope you have a wonderful day!')
        CALL function menu
    ELSE
        PRINT('Please enter a valid input!')
        CALL function customermenu
    ENDIF
ENDIF

```

Admin and super user login

```
DEFINE loginadmin
    DECLARE adminid, adminpassword
    DECLARE success AS BOOLEAN
    DECLARE adminlist AS LIST

    PRINT('Please enter your username:')
    INPUT adminid
    PRINT('Please enter your password:')
    INPUT adminpassword
    IF adminid EQUALS TO 'admin' and adminpassword EQUALS TO 'admin'
        CALL function supermenu
    ELSE
        success = False
        OPEN 'adminaccount.txt' AS adminaccount in READ MODE
        FOR i in adminaccount
            STRIP and SPLIT i USING ',' AS DELIMITER to adminlist
            IF (adminlist[0] EQUALS TO adminid and adminlist[1] EQUALS TO adminpassword)
                success = True
            ENDIF
        ENDFOR
        IF (success)
            PRINT('Admin login successful!')
            CALL function adminmenu
        ELSE
            PRINT('Invalid admin username and password!')
            PRINT('Please login again!')
            CALL function loginadmin
        ENDIF
    ENDIF
ENDIF
```

Admin's menu

```

DEFINE adminmenu
  DECLARE adminchoice, accounttype AS INTEGER

  PRINT('-----You are in admin menu-----')
  PRINT('=====')
  PRINT('Please choose an option 1.Create account 2.Edit customer detail 3.View all data 4.Transaction 5.Exit')
  INPUT adminchoice
  IF adminchoice = 1
    WHILE True
      PRINT('Which account do you want to create? 1.Saving 2.Current 3.Exit')
      INPUT accounttype
      IF accounttype = 1
        CALL function addSavingAccount
      ELSEIF accounttype = 2
        CALL function addCurrentAccount
      ELSEIF accounttype = 3
        CALL function adminmenu
        BREAK LOOP
      ELSE
        print('Please enter a valid input!')
      ENDIF
    ENDWHILE
  ELSEIF adminchoice = 2
    WHILE True
      PRINT('Which account do you want to edit 1.Saving 2.Current 3.Exit')
      INPUT accounttype
      IF accounttype = 1
        CALL function editsavingdetail
      ELSEIF accounttype = 2
        CALL function editcurrentdetail
      ELSEIF accounttype = 3
        CALL function adminmenu
        BREAK LOOP
      ELSE
        PRINT('Please enter a valid input!')
      ENDIF
    ENDWHILE
  ELSEIF adminchoice = 3
    CALL function viewcustomerdetail
  ELSEIF = 4
    CALL function transaction
  ELSEIF = 5
    PRINT('Thank you for using APU bank!')
    PRINT('Hope you have a wonderful day!')
    CALL fuction menu
  ELSE
    PRINT ('Please enter a valid input!')
    CALL function adminmenu
  ENDIF

```



Super user's menu

```
DEFINE supermenu
  DECLARE option AS INTEGER

  PRINT ('-----You are in super account menu-----')
  PRINT ('=====')
  PRINT ('Please choose an option 1.Create admin account 2.Check transaction 3.Exit')
  INPUT option
  IF option = 1
    CALL function addAdminAccount
  ELSEIF option = 2
    CALL fuction transaction
    break
  ELSEIF option = 3
    PRINT('Thank you for using APU bank!')
    PRINT('Hope you have a wonderful day!')
    CALL fuction menu
  ELSE
    PRINT 'Please enter a valid input!'
    CALL function supermenu
  ENDIF
```

Saving account registration

```
DEFINE addSavingAccount
    DECLARE name, phone_num, icnumber, balance, defaultpassword AS STRING
    DECLARE age, newAccNum, oldAccNum AS INTEGER
    DECLARE loginlist AS LIST

    WHILE True
        PRINT ('Please enter your name:')
        INPUT name
        PRINT('Name registered!')
        WHILE True
            PRINT('Please enter your age:')
            INPUT age
            IF age EQUALS TO INTEGER
                IF integer(age) <= 17
                    PRINT('You are not eligible to create an account')
                    CALL function adminmenu
                ELSE
                    PRINT('Age registered!')
                    BREAK LOOP
                ENDIF
            ELSE
                PRINT('Enter numbers only!')
            ENDIF
        ENDWHILE

        WHILE True
            PRINT('Please enter your phone number:')
            INPUT phone_num
            IF phone_num EQUALS TO INTEGER
                IF LENGTH of phone_num < 8
                    PRINT('Invalid phone number! Please enter again')
                ELSEIF LENGTH of phone_num >12
                    PRINT('Phone number too long! Please enter again')
                ELSE
                    PRINT('Phone number registered')
                    BREAK LOOP
                ENDIF
            ELSE
                PRINT('Enter numbers only!')
            ENDIF
        ENDWHILE
    ENDWHILE
```

```

WHILE True
    PRINT('Please enter your IC number:')
    INPUT icnumber
    IF icnumber EQUALS TO INTEGER
        IF LENGTH of icnumber > 12
            PRINT('IC number more than 12 digit, please enter again')
        ELSEIF LENGTH of icnumber < 12
            PRINT('IC number less than 12 digit, please enter again')
        ELSE
            PRINT('IC number registered')
            BREAK LOOP
        ENDIF
    ELSE
        PRINT('Enter numbers only!')
    ENDIF
ENDWHILE

OPEN 'Customerlogin.txt' AS loginfile in READ MODE
    balance = 0
    FOR line in loginfile
        STRIP and SPLIT line USING ',' AS DELIMITER TO loginlist
        PASS to LAST line
    ENDFOR
    oldAccNum = loginlist[0]
    newAccNum = integer(oldAccNum) + 1
OPEN 'Customerlogin.txt' AS loginfile in APPEND MODE
    defaultpassword = string('default')
    loginfile.write('\n')
    DECALRE newAccNum
    loginfile.write(string(newAccNum) + ',' + string(defaultpassword) + ',' + 's')
OPEN 'Savinginfo.txt' AS loginfile in APPEND MODE
    savinginfo.write('\n')
    savinginfo.write('name + '|' + age + '|' + phone_num + '|' + icnumber + '|' + string(newAccNum) + '|' + string(balance))
    PRINT('Account has been made!')
    PRINT('Your account info:' + '|' + 'name + '|' + age + '|' + phone_num + '|' + icnumber)
    PRINT('This is your account number:' + str(newAccNum))
    PRINT('Your password is"default", Please change immediately!')
    CALL function adminmenu

```

Current account registration

```
DEFINE addCurrentAccount
    DECLARE name, phone_num, icnumber, balance, defaultpassword AS STRING
    DECLARE age, newAccNum, oldAccNum AS INTEGER
    DECLARE loginlist AS LIST

    WHILE true
        PRINT ('Please enter your name:')
        INPUT name
        PRINT('Name registered!')
        WHILE True
            PRINT('Please enter your age:')
            INPUT age
            IF age EQUALS TO INTEGER
                IF integer(age) <= 17
                    PRINT('You are not eligible to create an account')
                    CALL function adminmenu
                ELSE
                    PRINT('Age registered!')
                    BREAK LOOP
                ENDIF
            ELSE
                PRINT('Enter numbers only!')
            ENDIF
        ENDWHILE

        WHILE True
            PRINT('Please enter your phone number:')
            INPUT phone_num
            IF phone_num EQUALS TO INTEGER
                IF LENGTH of phone_num < 8
                    PRINT('Invalid phone number! Please enter again')
                ELSEIF LENGTH of phone_num >12
                    PRINT('Phone number too long! Please enter again')
                ELSE
                    PRINT('Phone number registered')
                    BREAK LOOP
                ENDIF
            ELSE
                PRINT('Enter numbers only!')
            ENDIF
        ENDWHILE
    ENDWHILE
```

```

WHILE True
    PRINT('Please enter your IC number:')
    INPUT icnumber
    IF icnumber EQUALS TO INTEGER
        IF LENGTH of icnumber > 12
            PRINT('IC number more than 12 digit, please enter again')
        ELSEIF LENGTH of icnumber < 12
            PRINT('IC number less than 12 digit, please enter again')
        ELSE
            PRINT('IC number registered')
            BREAK LOOP
        ENDIF
    ELSE
        PRINT('Enter numbers only!')
    ENDIF
ENDWHILE

OPEN 'Customerlogin.txt' AS loginfile in READ MODE
    balance = 0
    FOR line in loginfile
        STRIP and SPLIT i USING ',' AS DELIMITER TO loginlist
        PASS TO LAST line
    ENDFOR
    oldAccNum = loginlist[0]
    newAccNum = int(oldAccNum) + 1
OPEN 'Customerlogin.txt' AS loginfile in APPEND MODE
    defaultpassword = string('default')
    loginfile.write('\n')
    loginfile.write(string(newAccNum) + ',' + string(defaultpassword) + ',' + 'C')
OPEN 'Currentinfo.txt' AS loginfile in APPEND MODE
    loginfile.write('\n')
    loginfile.write('name + '|' + age + '|' + phone_num + '|' + icnumber + '|' + string(newAccNum) + '|' + string(balance))
    PRINT('Account has been made!')
    PRINT('Your account info:' + '|' + 'name + '|' + age + '|' + phone_num + '|' + icnumber)
    PRINT('This is your account number:' + string(newAccNum))
    PRINT('Your password is"default", Please change immediately!')
    CALL function adminmenu

```

Admin account registration

```
DEFINE addAdminAccount
  DECLARE adminid, adminpassword, confirmpassword AS STRING

  PRINT ('Please create username:')
  INPUT adminid
  PRINT ('Please create password:')
  INPUT adminpassword
  PRINT ('Please confirm password:')
  INPUT confirmpassword
  IF adminpassword NOT EQUAL confirmpassword
    PRINT ('Password does not match!')
    CALL function addAdminAccount
  ELSE
    IF LENGTH of adminpassword <= 7
      PRINT ('Password is too short!')
    ELSE
      OPEN 'adminaccount.txt' AS adminaccount in APPEND MODE
      WRITE adminid and adminpassword in adminaccount
      PRINT ('Admin account has been created!')
      CALL function supermenu
    ENDIF
  ENDIF
```



### Changing customer password

```
DEFINE changecustomerpassword
  DECLARE newpassword AS STRING
  DECLARE allrecord,reclist AS LIST
  DECLARE all AS INTEGER

  OPEN 'Customerlogin.txt' AS customerlogin in READ MODE
  FOR rec in customerlogin
    STRIP and SPLIT rec USING ',' AS DELIMITER to reclist
    allrecord.append(reclist)
  ENDFOR
  PRINT('Please enter your new password: ')
  INPUT newpassword
  i = -1
  all = LENGTH of allrecord
  FOR count in range(0,all)
    IF loginlist[0] EQUALS TO allrecord[count][0]
      i = count
    ENDIF
  ENDFOR
  IF i > 0
    allrecord[i][1] = newpassword
    BREAK LOOP
  ENDIF
  OPEN 'Customerlogin.txt' AS customerlogin in WRITE MODE
  all = LENGTH of allrecord
  FOR count in range(0,all)
    rec = JOIN(all[count]) with ',' + '\n'
    customerlogin.write(rec)
  ENDFOR
  PRINT('Password has been changed successfully')
  PRINT('Please login again!')
  CALL function logincustomer
```

Edit saving customer detail

```

DEFINE editsavingdetail
  DECLARE allrec ,reclist, editedlist AS LIST
  DECLARE accountnumber, editchoice, newage, newphonenum, joinstring AS STRING

  OPEN 'Savinginfo.txt' AS saving in READ MODE
  PRINT('Please enter account number to edit:')
  INPUT accountnumber
  num = 0
  FOR rec in saving
    STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
    allrec.append(reclist)
  ENDFOR

  OPEN 'Savinginfo.txt' AS saving in READ MODE
  FOR rec in saving
    STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
    IF accountnumber EQUALS TO reclist[4]
      PRINT('Account found!')
      PRINT('Account:' + string(reclist))
    ENDFOR

    while True
      PRINT('Choose a detail to edit 1.Age 2.Phone number 3.Save and exit')
      INPUT editchoice
      IF editchoice = 1
        PRINT('Please enter new age:')
        INPUT newage
        reclist[1] = newage
        PRINT('Age has been changed successfully!')
        PRINT('New customer detail:' + string(reclist))
        allrec[num] = reclist
        CONTINUE LOOP
      ELSEIF editchoice = 2
        PRINT('Please enter new phone number:')
        INPUT newphonenum
        reclist[2] = newphonenum
        PRINT('Phone number has been changed successfully!')
        PRINT('New customer detail:' + string(reclist))
        CONTINUE LOOP
      ELSEIF editchoice = 3
        OPEN 'Savinginfo.txt' AS blank in WRITE MODE
        blank.write('')
        OPEN 'Savinginfo.txt' AS saving in APPEND MODE
        FOR edited_list in allrec
          joinstring = '|'.join(edited_list)
          saving.write(joinstring + '\n')
        ENDFOR
        CALL function adminmenu
        BREAK LOOP
      ELSE
        PRINT('Please enter a valid input!')
      ENDIF
    ENDWHILE
  ELSE
    PRINT('Account number does not exist!')
  ENDIF
  num = num + 1

```



Edit current customer detail

```

DEFINE editcurrentdetail
    DECLARE allrec ,reclist, AS LIST
    DECLARE accountnumber, editchoice, newage, newphonenumber, joinstring AS STRING
    DECLARE num AS INTEGER

    OPEN 'Currentinfo.txt' AS current in READ MODE
        PRINT('Please enter account number to edit:')
        INPUT accountnumber
        num = 0
        FOR rec in current
            STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
            allrec.append(reclist)
        ENDFOR

    OPEN 'Currentinfo.txt' AS current in READ MODE
        FOR rec in current
            STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
            IF accountnumber EQUALS TO reclist[4]
                PRINT('Account found!')
                PRINT('Account:' + string(reclist))
            ENDFOR

            while True
                PRINT('Choose a detail to edit 1.Age 2.Phone number 3.Save and exit')
                INPUT editchoice
                IF editchoice = 1
                    PRINT('Please enter new age:')
                    INPUT newage
                    reclist[1] = newage
                    PRINT('Age has been changed successfully!')
                    PRINT('New customer detail:' + string(reclist))
                    allrec[num] = reclist
                    CONTINUE
                ELSEIF editchoice = 2
                    PRINT('Please enter new phone number:')
                    INPUT newphonenumber
                    reclist[2] = newphonenumber
                    PRINT('Phone number has been changed successfully!')
                    PRINT('New customer detail:' + string(reclist))
                    CONTINUE
                ELSEIF editchoice = 3
                    OPEN 'Currentinfo.txt' AS blank in WRITE MODE
                    blank.write('')
                    OPEN 'Currentinfo.txt' AS saving in APPEND MODE
                    FOR edited_list in allrec
                        joinstring = '|'.join(edited_list)
                        saving.write(joinstring + '\n')
                    ENDFOR
                    CALL function adminmenu
                    BREAK LOOP
                ELSE
                    PRINT('Please enter a valid input!')
                ENDFOR
            ENDWHILE
        ELSE
            PRINT('Account number does not exist!')
        ENDFOR
        num = num + 1

```

## Saving account withdrawal

```

DEFINE savingwithdrawal
  DECLARE allrec, reclist AS LIST
  DECLARE newbalance, withdrawamount AS INTEGER
  DECLARE joinstring, tran AS STRING

  PRINT('-----Withdraw from savings-----')
  PRINT('=====')
  OPEN 'Savinginfo.txt' AS saving in READ MODE
  FOR rec in saving
    STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
    allrec.append(reclist)
    IF loginlist[0] EQUALS TO reclist[4]
      PRINT('Your balance is:RM' + string(reclist[5]))
      TRY
        PRINT('Enter withdraw amount:')
        INPUT integer(withdrawamount)
      ENDTRY
      EXCEPT ValueError
        PRINT(('Enter numbers only!'))
      ENDEXCEPT
      new balance = integer(reclist[5]) - integer(withdrawamount)
      IF integer(newbalance) < 100
        PRINT('Withdraw unsuccessful!')
        PRINT('Balance insufficient!')
      ELSE
        reclist[5] = str(newbalance)
        PRINT('Your balance is :RM + reclist[5])
      ENDIF

      date = datetime.datetime.now()
      OPEN 'Transaction.txt' AS transactionfile in APPEND MODE
      tran = (reclist[0] + ',' + reclist[4] + ',' + 'withdraw' + ',' + 'RM' + string(withdrawamount) + ',' + date.strftime('%d/%m/%Y') + '\n')
      transactionfile.write(tran)
    ENDIF
  ENDFOR

  OPEN 'Savinginfo.txt' AS blank in WRITE MODE
  blank.write('')
  OPEN 'Savinginfo.txt' AS saving in APPEND MODE
  FOR newrec in allrec
    joinstring = '|'.join(newrec)
    saving.write(joinstring + '\n')
  ENDFOR

```

## Current account withdrawal

```

DEFINE currentwithdrawal
  DECLARE allrec, reclist AS LIST
  DECLARE newbalance, withdrawamount AS INTEGER
  DECLARE joinstring, tran AS STRING

  PRINT('-----Withdraw from current-----')
  PRINT('=====')
  OPEN 'Currentinfo.txt' AS current in READ MODE
  FOR rec in current
    STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
    allrec.append(reclist)
    IF loginlist[0] EQUALS TO reclist[4]
      PRINT('Your balance is:RM' + string(reclist[5]))
      TRY
        PRINT('Enter withdraw amount:')
        INPUT integer(withdrawamount)
      ENDTRY
      EXCEPT ValueError
        PRINT(('Enter numbers only!'))
      ENDEXCEPT
      new balance = integer(reclist[5]) - integer(withdrawamount)
      IF integer(newbalance) < 100
        PRINT('Withdraw unsuccessful!')
        PRINT('Balance insufficient!')
      ELSEIF integer(newbalance) >= 100
        reclist[5] = str(newbalance)
        PRINT('Your balance is :RM + reclist[5])
      ENDIF

      date = datetime.datetime.now()
      OPEN 'Transaction.txt' AS transactionfile in APPEND MODE
      tran = (reclist[0] + ',' + reclist[4] + ',' + 'withdraw' + ',' + 'RM' + string(withdrawamount) + ',' + date.strftime('%d/%m/%Y') + '\n')
      transactionfile.write(tran)
    ENDIF
  ENDFOR

  OPEN 'Currentinfo.txt' AS blank in WRITE MODE
  blank.write('')
  OPEN 'Currentinfo.txt' AS saving in APPEND MODE
  FOR newrec in allrec
    joinstring = '|'.join(newrec)
    saving.write(joinstring + '\n')
  ENDFOR

```

## Saving account deposit

```

DEFINE savingdeposit
    DECLARE allrec,reclist , AS LIST
    DECLARE newbalance, depositamount, AS INTEGER
    DECLARE joinstring, tran AS STRING

    PRINT('-----Deposit to Savings-----')
    PRINT('=====')
    OPEN 'Savinginfo.txt' AS saving in READ MODE
    FOR rec in saving
        STRIP and SPLIT rec USING '|' AS DELIMITER To reclist
        allrec.append(reclist)
        IF loginlist[0] EQUALS TO reclist [4]
            PRINT('Your balance is:RM + string(reclist[5])
            TRY
                PRINT('Enter deposit amount')
                INPUT integer(depositamount)
            ENDTRY
            EXCEPT ValueError
                PRINT('Enter numbers only!')
            ENDEXCEPT
            newbalance = integer(reclist[5]) + interger(depositamount)
            reclist[5] = string(newbalance)
            PRINT('Your balance is: RM' + RECLIST[5])

            date = datetime.datetime.now()
            OPEN 'Transaction.txt' AS transactionfile in APPEND MODE
            tran = (reclist[0] + ',' + reclist[4] + ',' + 'deposit' + ',' + 'RM'+ string(depositamount) + ',' + date.strftime('%d:%m:%Y') + '\n')
            transactionfile.write(tran)

        ENDIF
    ENDFOR

    OPEN 'Savinginfo.txt' AS blank in WRITE MODE
    blank.write('')
    OPEN 'Savinginfo.txt' AS saving in APPEND MODE
    FOR newrec in allrec
        joinstring = '|'.join(newrec)
        saving.write(joinstring + '\n')

```

## Current account deposit

```

DEFINE currentdeposit
    DECLARE allrec,reclist , AS LIST
    DECLARE newbalance, depositamount, AS INTEGER
    DECLARE joinstring AS STRING

    PRINT('-----Deposit to current-----')
    PRINT('=====')
    OPEN 'Currentinfo.txt' AS current in READ MODE
    FOR rec in current
        STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
        allrec.append(reclist)
        IF loginlist[0] EQUALS TO reclist [4]
            PRINT('Your balance is:RM + string(reclist[5])
            TRY
                PRINT('Enter deposit amount')
                INPUT integer(depositamount)
            ENDTRY
            EXCEPT ValueError
                PRINT('Enter numbers only!')
            ENDEXCEPT
            newbalance = integer(reclist[5]) + interger(depositamount)
            reclist[5] = string(newbalance)
            PRINT('Your balance is: RM' + RECLIST[5])

            date = datetime.datetime.now()
            OPEN 'Transaction.txt' AS transactionfile in APPEND MODE
            tran = (reclist[0] + ',' + reclist[4] + ',' + 'deposit' + ',' + 'RM'+ string(depositamount) + ',' + date.strftime('%d:%m:%Y') + '\n')
            transactionfile.write(tran)

        ENDIF
    ENDFOR

    OPEN 'Currentinfo.txt' AS blank in WRITE MODE
    blank.write('')
    OPEN 'Currentinfo.txt' AS current in APPEND MODE
    FOR newrec in allrec
        joinstring = '|'.join(newrec)
        saving.write(joinstring + '\n')

```

Balance inquiry for customers

```

DEFINE balanceinquiry
    DECALRE available AS INTEGER

    OPEN 'Savinginfo.txt' AS saving in READ MODE
    FOR rec in saving
        STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
        IF loginlist[0] EQUALS TO reclist[4]
            available = integer(reclist[5]) - 100
            PRINT('Your balance is :RM + reclist[5])
            PRINT('Maximum withdrawal is:RM' + str(available))
        ENDIF
    ENDFOR
    OPEN 'Currentinfo.txt' AS current in READ MODE
    FOR rec in current
        STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
        IF loginlist[0] EQUALS TO reclist[4]
            available = integer(reclist[5]) - 500
            PRINT('Your balance is :RM + reclist[5])
            PRINT('Maximum withdrawal is:RM' + str(available))
        ENDIF
    ENDFOR
    CALL function customermenu

```

View customers' detail for admins only

```

DEFINE viewcustomerdetail
    DECLARE admininput AS INTEGER

    PRINT('Which account detail do you want to view? 1.Saving 2.Current 3.Exit:)
    INPUT admininput
    IF admininput = 1
        PRINT(' Name |Age|Phone no| IC number |Acc no|Balance')
        PRINT('=====')
        OPEN 'Savinginfo.txt' AS saving in READ MODE
        FOR line in saving
            PRINT(line)
        ENDFOR
        CALL function viewcustomerdetail
    ELSEIF admininput = 2
        OPEN 'Currentinfo.txt' AS current in READ MODE
        PRINT(' Name |Age|Phone no| IC number |Acc no|Balance')
        PRINT('=====')
        FOR line in current
            PRINT(line)
        ENDFOR
        CALL function viewcustomerdetail
    ELSEIF admininput = 3
        CALL function adminmenu
    ELSE
        PRINT('Please enter a valid input!)
        CALL fuction viewcustomerdetail
    ENDIF

```

## View customer transaction

```

DEFINE transaction
    DECLARE accountnumber, startday, startmonth, startyear, endday, endmonth, endyear, startdate, enddate AS STRING
    DECLARE i, count, totalwithdrawal, totaldeposit AS INTEGER
    DECLARE allrec AS LIST

    OPEN 'Transaction.txt' AS transactionfile in READ MODE
    FOR tran in transactionfile
        STRIP and SPLIT rec USING '|' AS DELIMITER to reclist
        allrec.append(reclist)
    ENDFOR

    PRINT('Please enter your account number: ')
    INPUT accountnumber
    WHILE True
        TRY
            PRINT('Please enter start day:')
            INPUT startday
            PRINT('Please enter start month:')
            INPUT startmonth
            PRINT('Please enter start year:')
            INPUT startyear
            datetime.datetime.strptime(startday, '%d')
            datetime.datetime.strptime(startmonth, '%m')
            datetime.datetime.strptime(startyear, '%y')
            BREAK LOOP
        ENDTRY
        EXCEPT ValueError
            PRINT('Please follow the format (dd for day) (mm for month) (yyyy for year)')
        ENDEXCEPT
    ENDWHILE

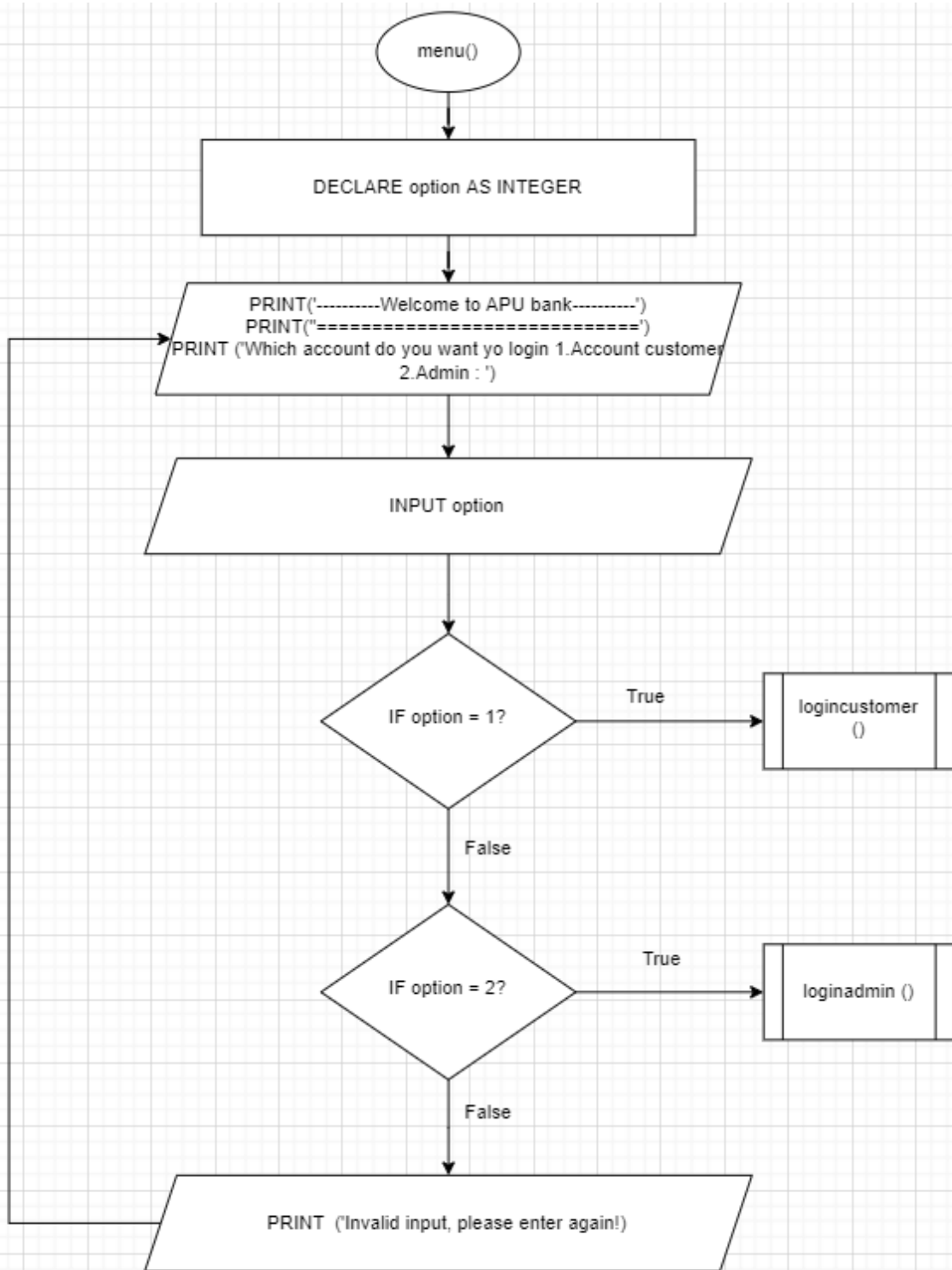
    WHILE True
        TRY
            PRINT('Please enter end day:')
            INPUT endday
            PRINT('Please enter end month:')
            INPUT endmonth
            PRINT('Please enter end year:')
            INPUT endyear
            datetime.datetime.strptime(endday, '%d')
            datetime.datetime.strptime(endmonth, '%m')
            datetime.datetime.strptime(endyear, '%y')
            BREAK LOOP
        ENDTRY
        EXCEPT ValueError
            PRINT('Please follow the format (dd for day) (mm for month) (yyyy for year)')
        ENDEXCEPT
    ENDWHILE

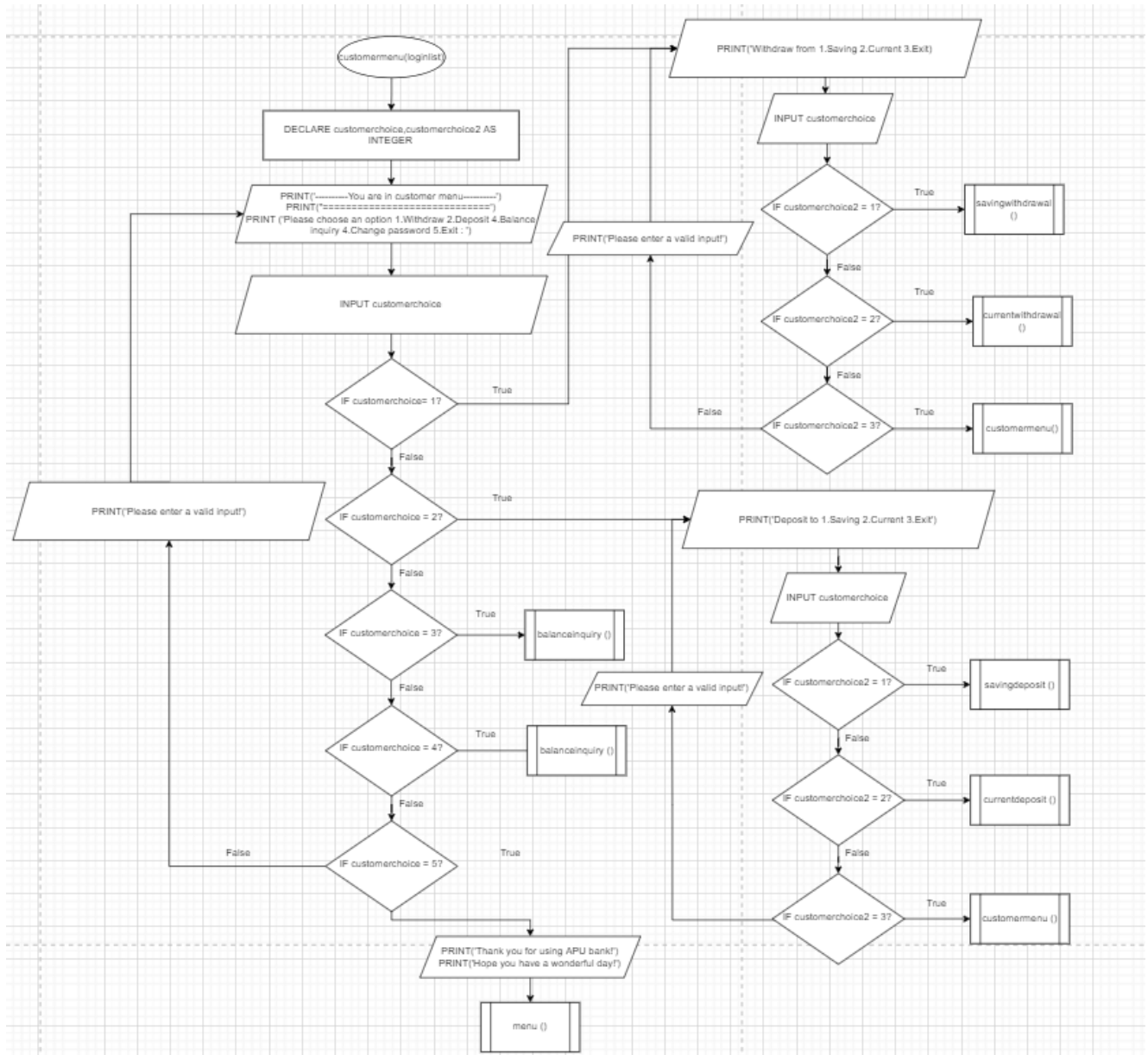
    IF LENGTH of startday = 1
        startday = '0' + string(startday)
    ENDIF
    IF LENGTH of endday = 1
        endday = '0' + string(endday)
    ENDIF
    IF LENGTH of startmonth = 1
        startmonth = '0' + string(startmonth)
    ENDIF
    IF LENGTH of endmonth = 1
        endmonth = '0' + string(endmonth)
    ENDIF

    startdate = startday + "/" + startmonth + "/" + startyear
    enddate = endday + "/" + endmonth + "/" + endyear
    PRINT(startdate)
    PRINT(enddate)
    totalwithdrawal = 0
    totaldeposit = 0
    i = -1

```

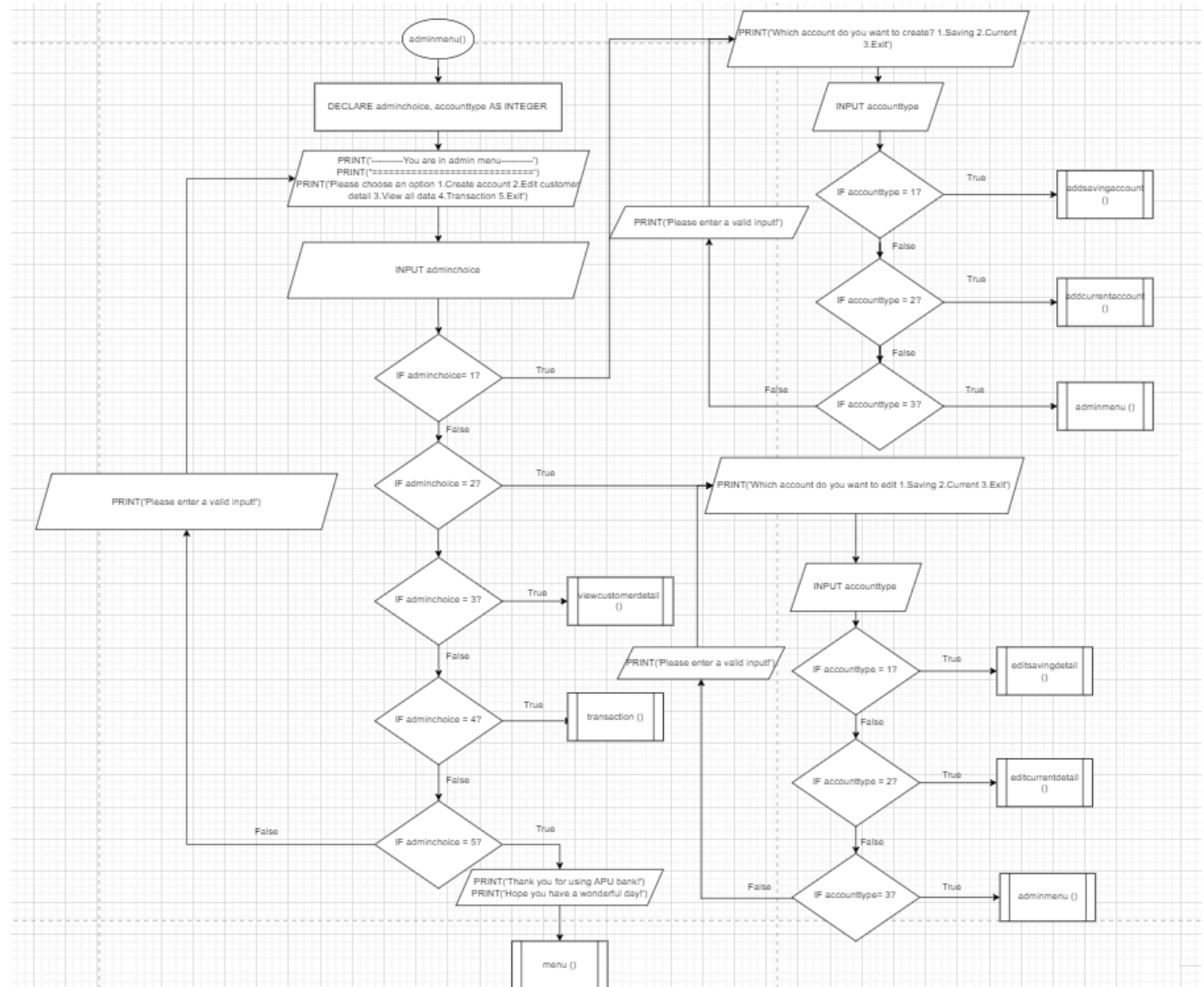
```
totalrec = LENGTH of (allrec)
FOR count in RANGE(0 TO totalrec)
    IF accountnumber EQUALS to allrec[count][1]
        i = count
        IF startdate <= (allrec[i][4]) AND enddate >= (allrec[i][4])
            IF allrec[i][2] EQUALS to 'withdraw'
                num = integer(allrec[i][3][2:])
                totalwithdraw += num
                PRINT('allrec[i]')
            ENDIF
            IF allrec[i][2] == 'deposit':
                num = integer(allrec[i][3][2:])
                totaldeposit += num
                PRINT(allrec[i])
            ENDIF
        ENDIF
    ENDIF
ENDFOR
IF i >= 0
    PRINT('Total withdrawal: ' + 'RM' + string(totalwithdrawal))
    PRINT(;;Total deposit: ' + 'RM' + string(totaldeposit))
ENDIF
CALL function adminmenu
:
```

**Flowchart****Menu**

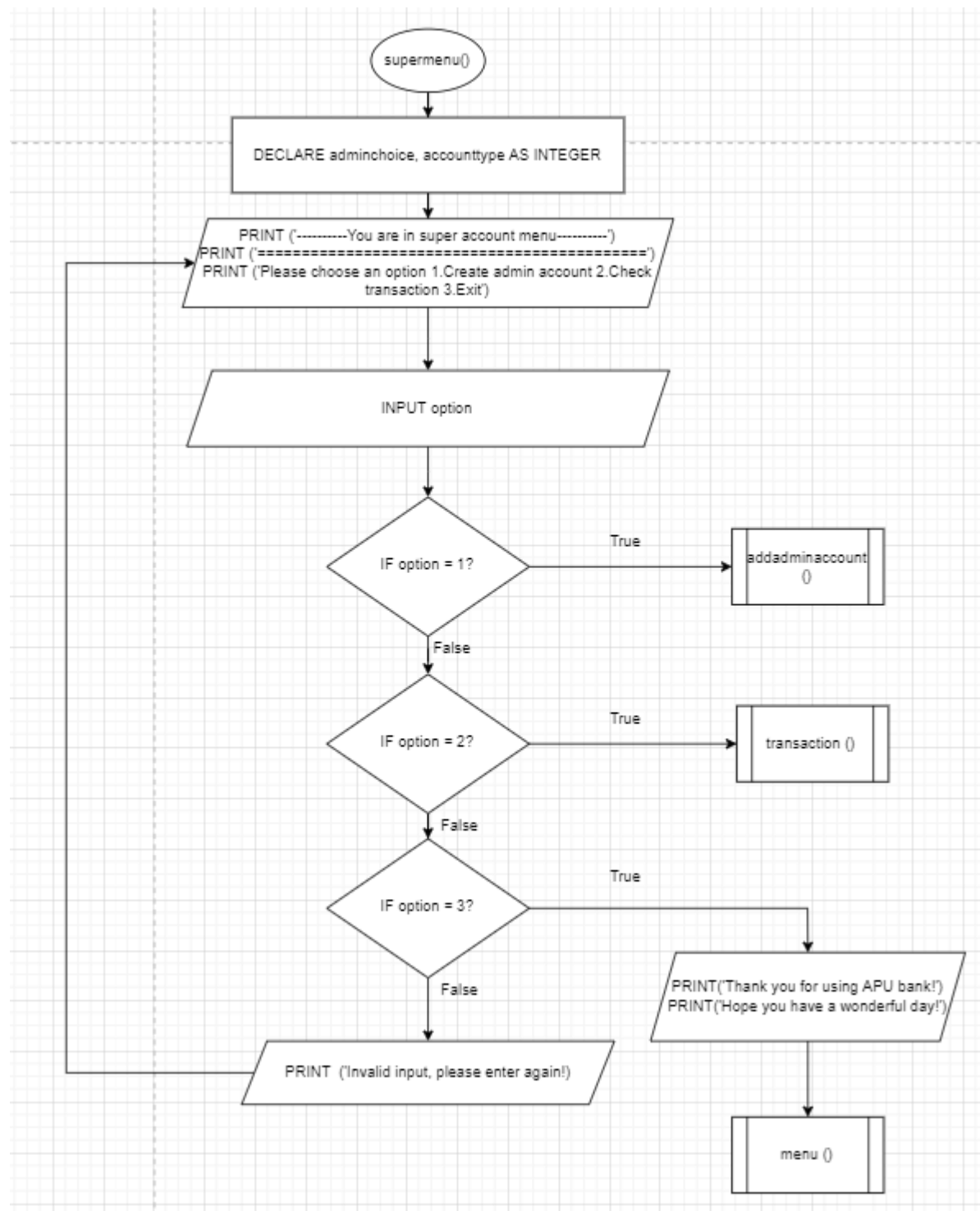
**customer menu**

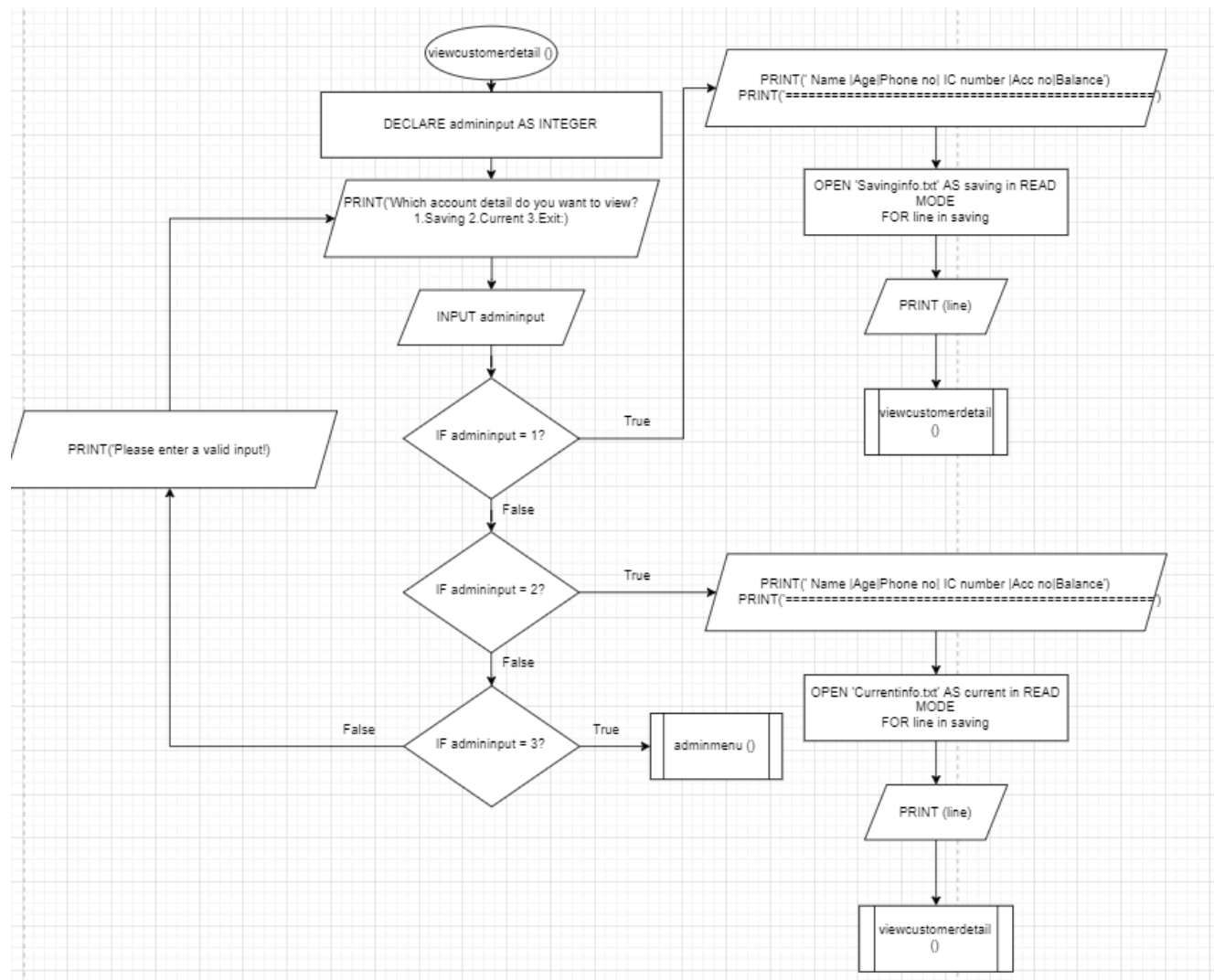


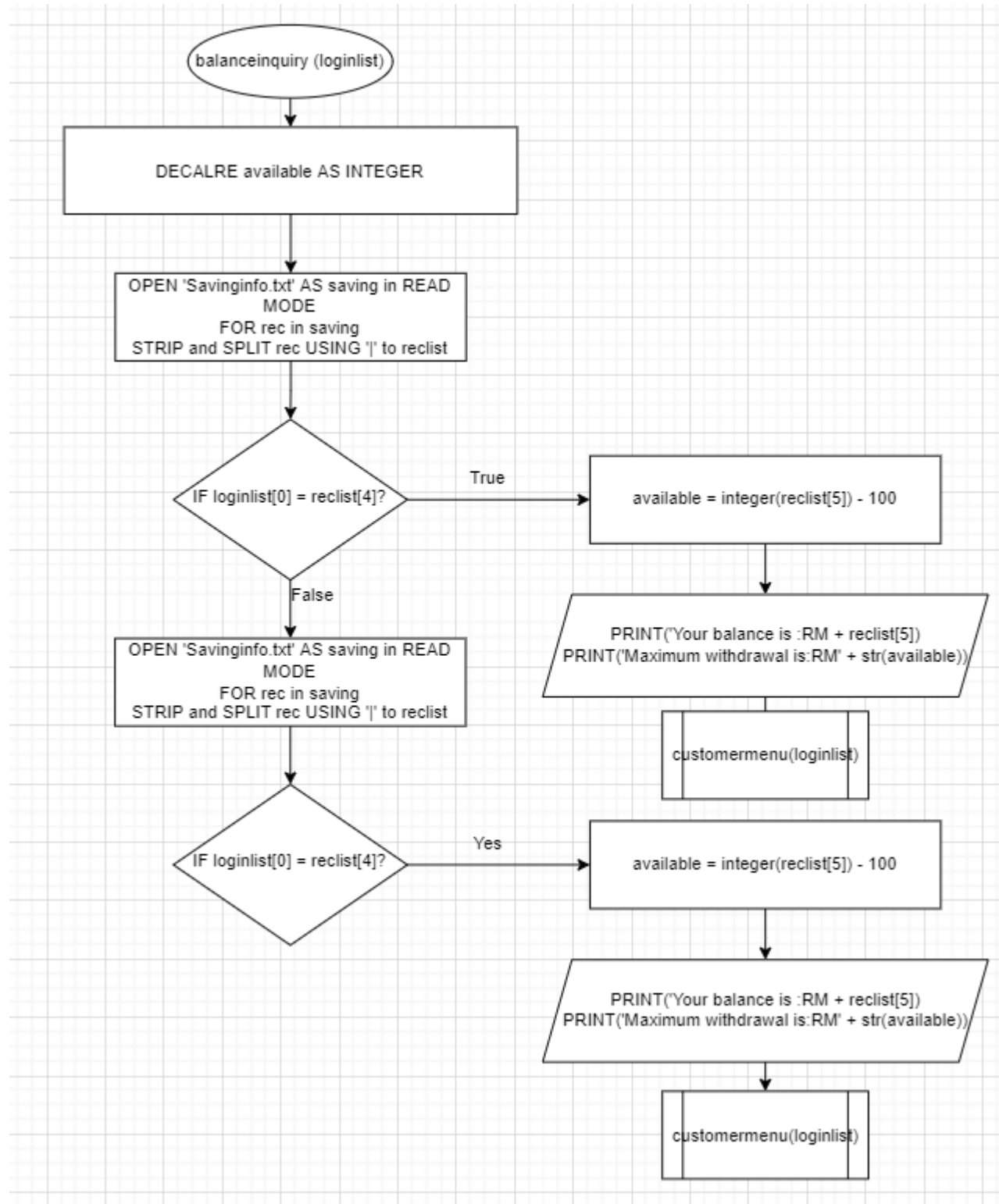
## Admin menu



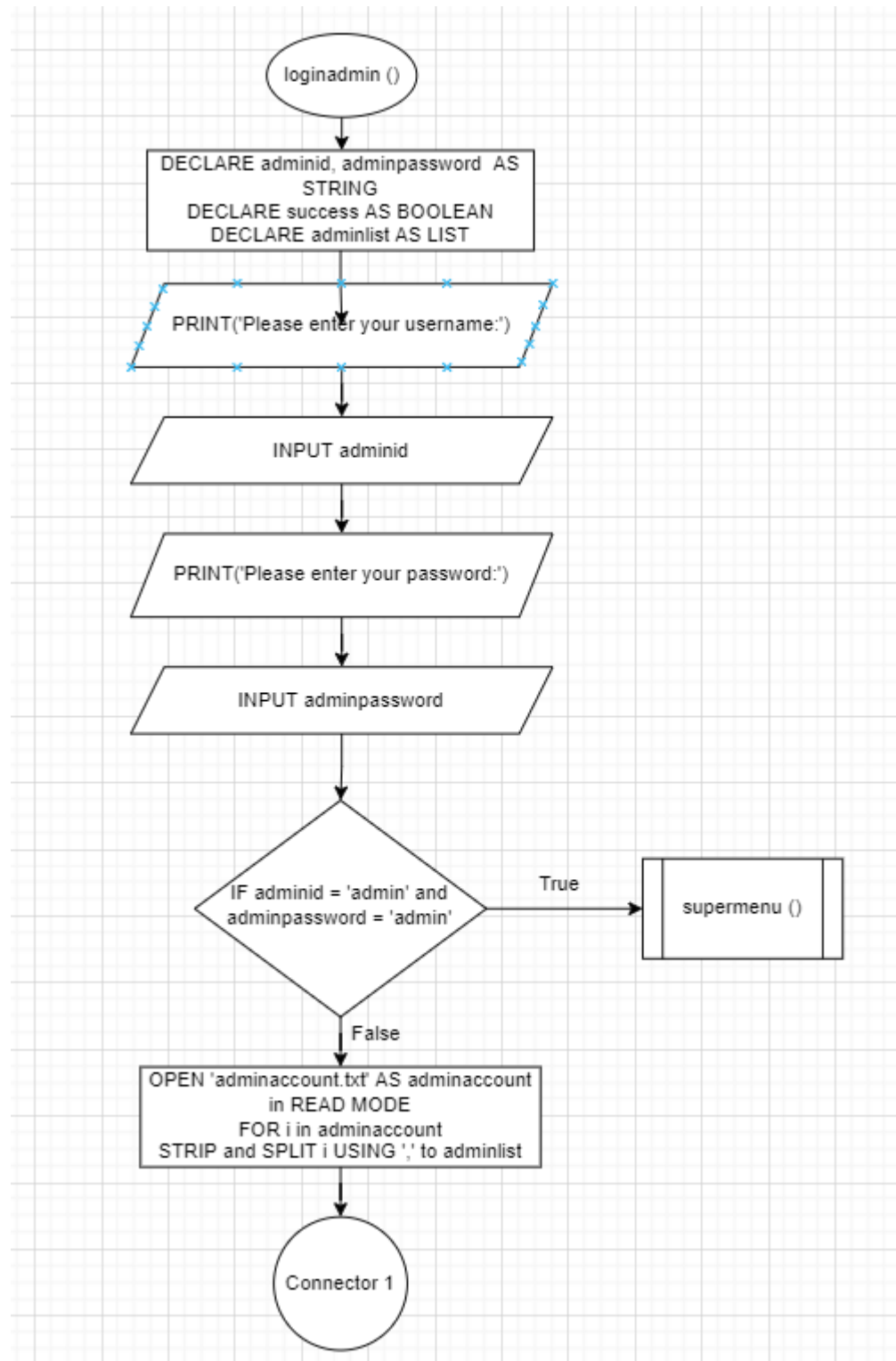
## Super user menu

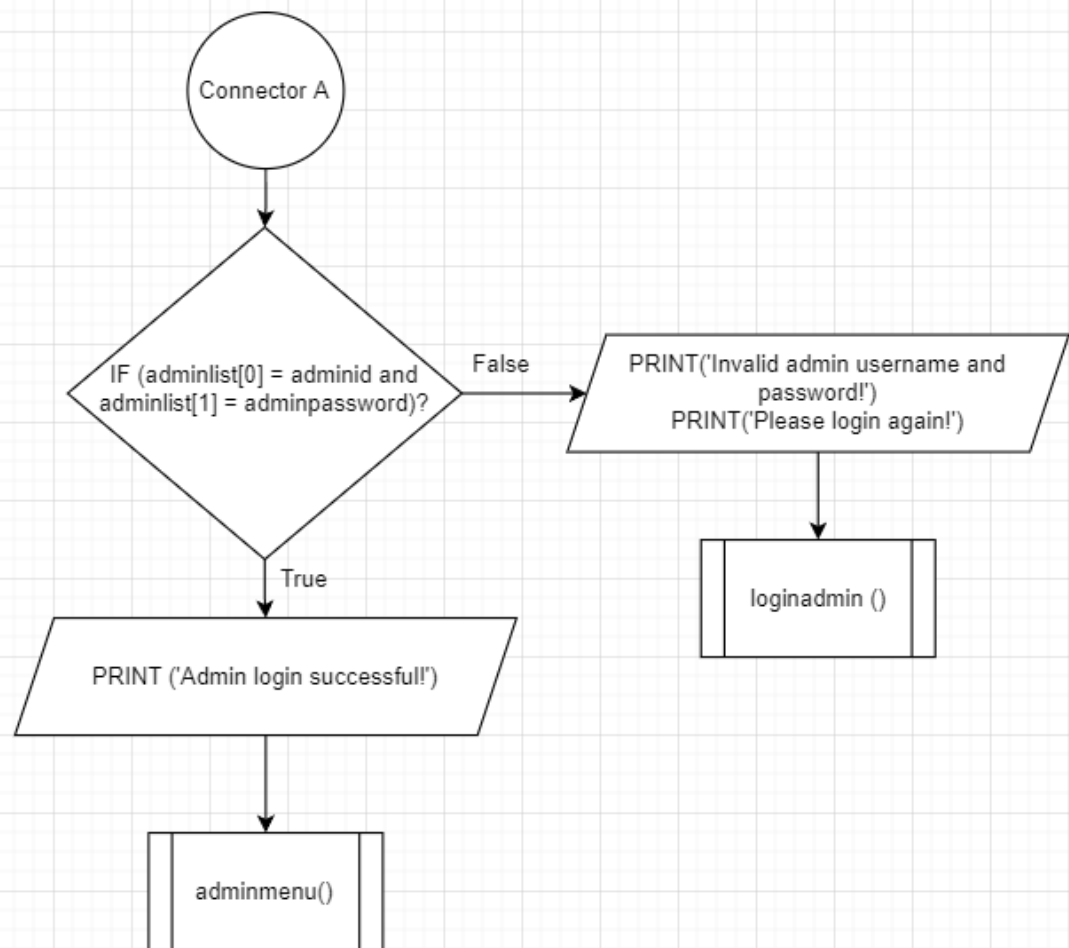


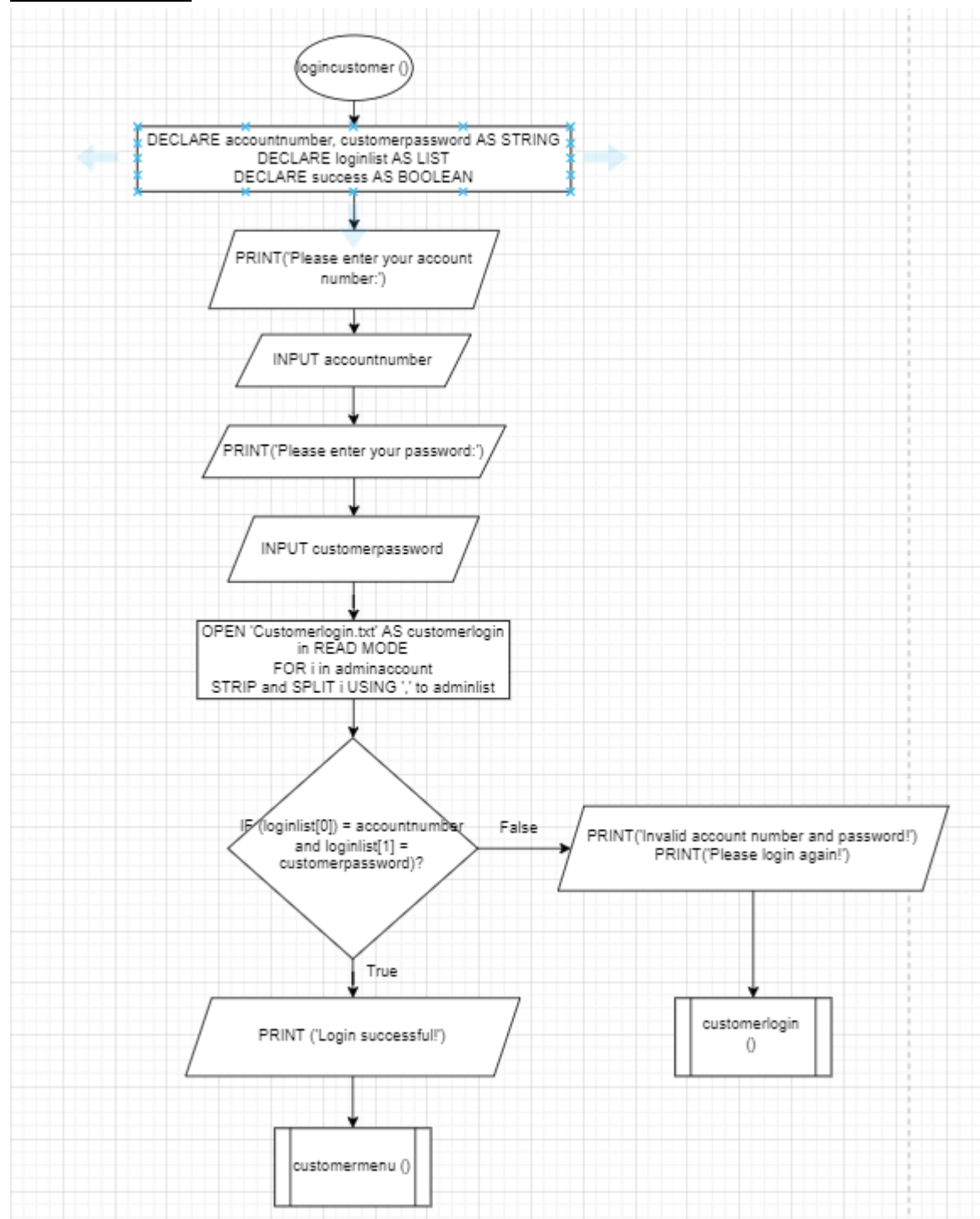
**View customer detail****Balance inquiry**

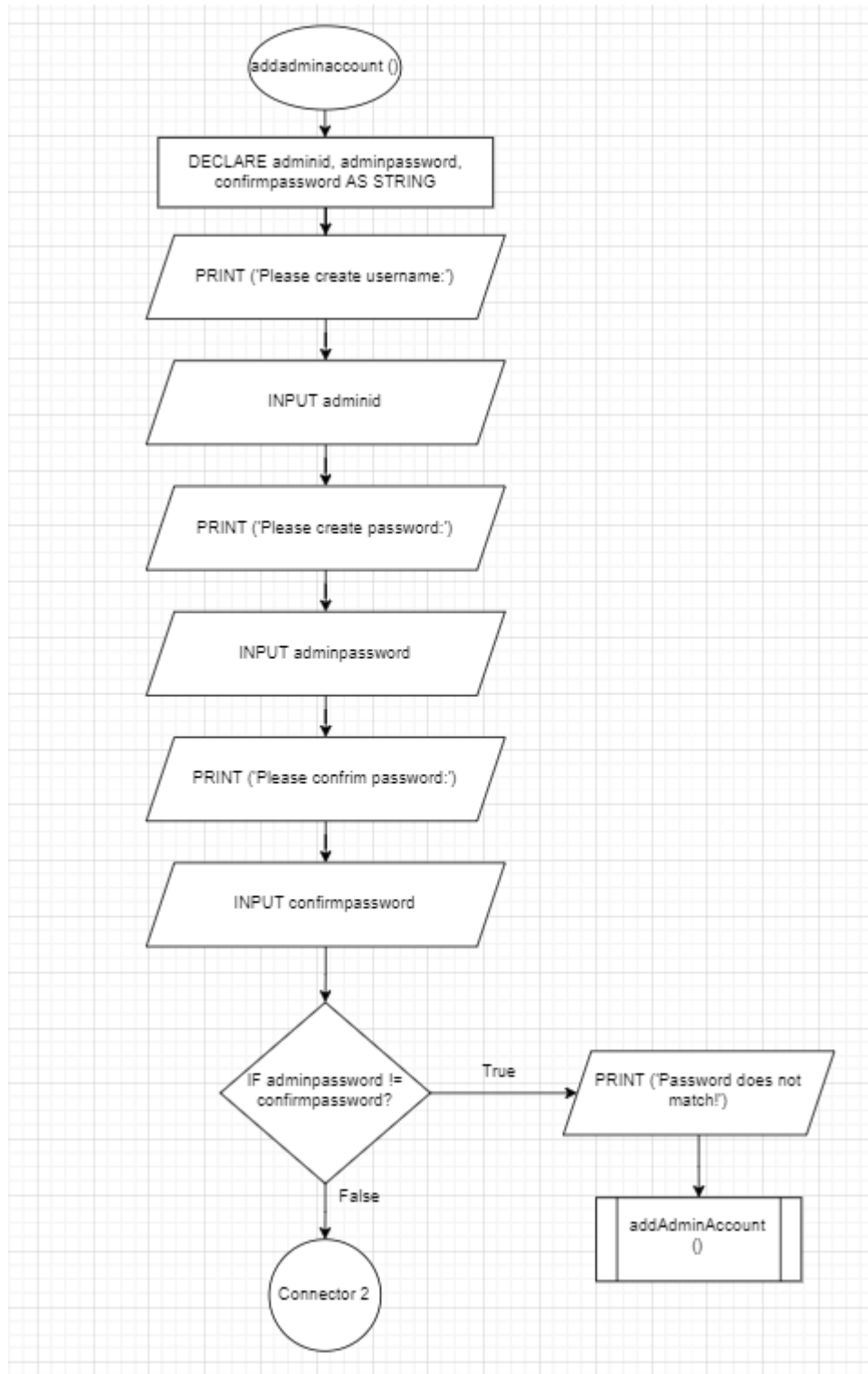


### Login admin

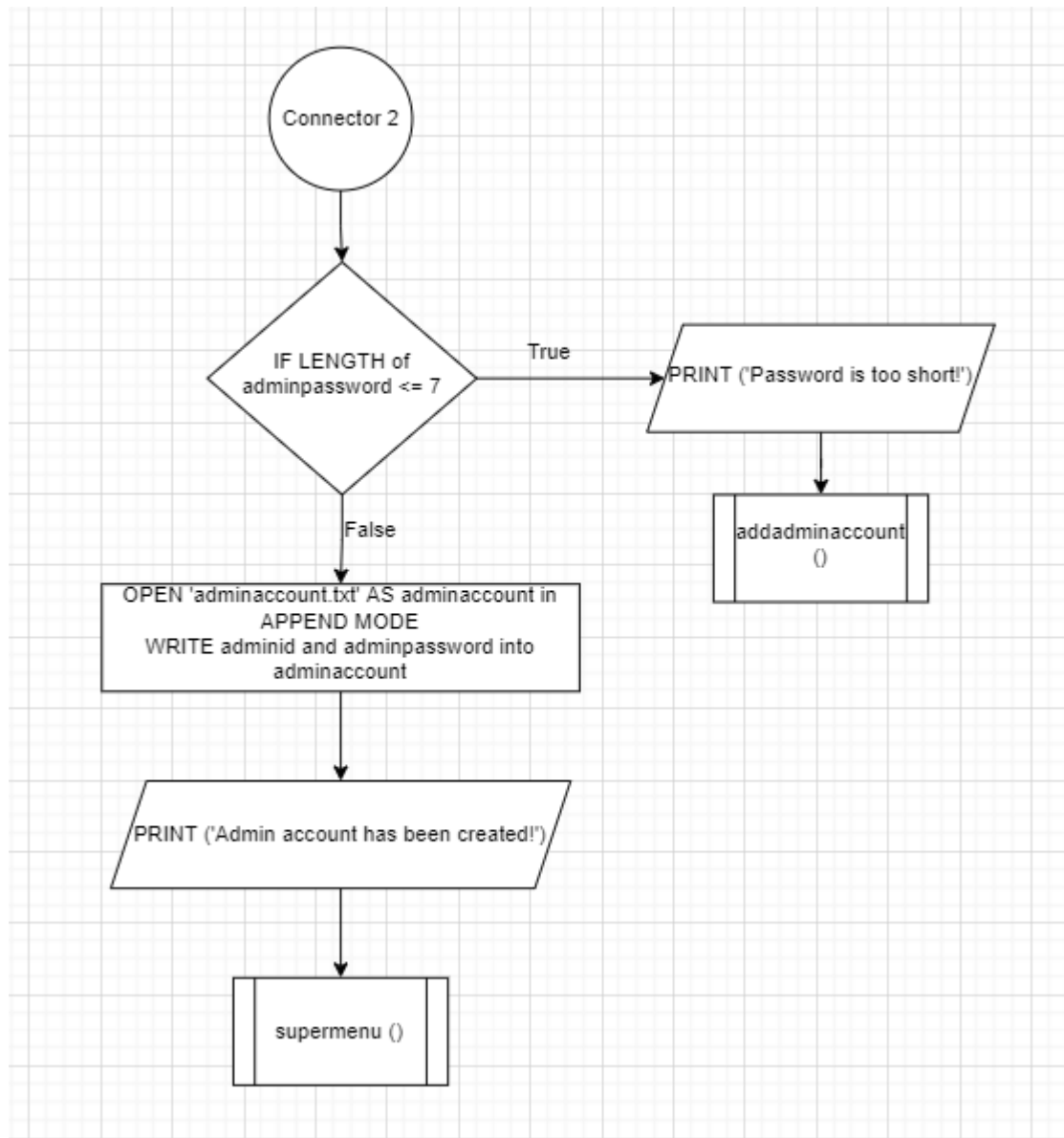


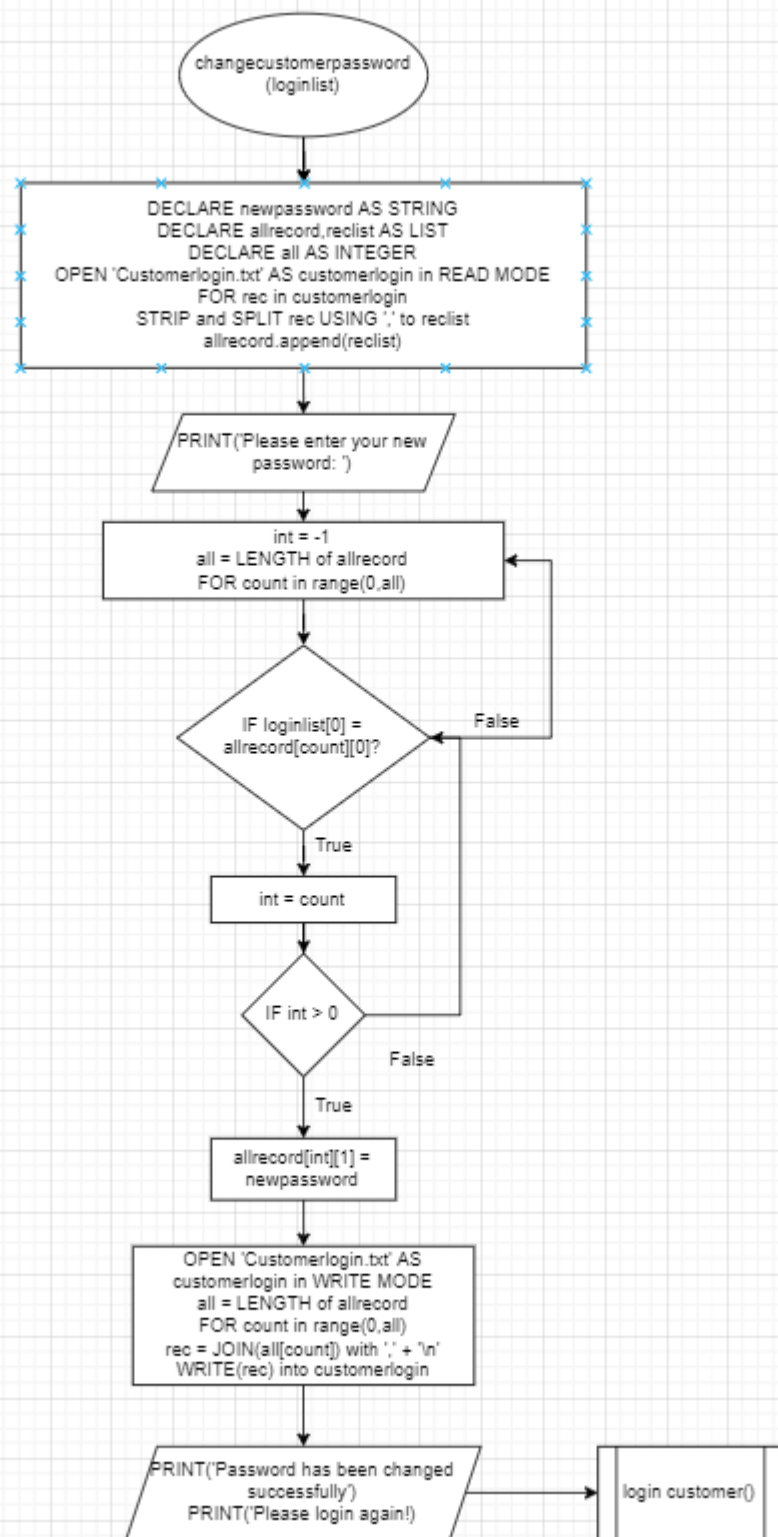


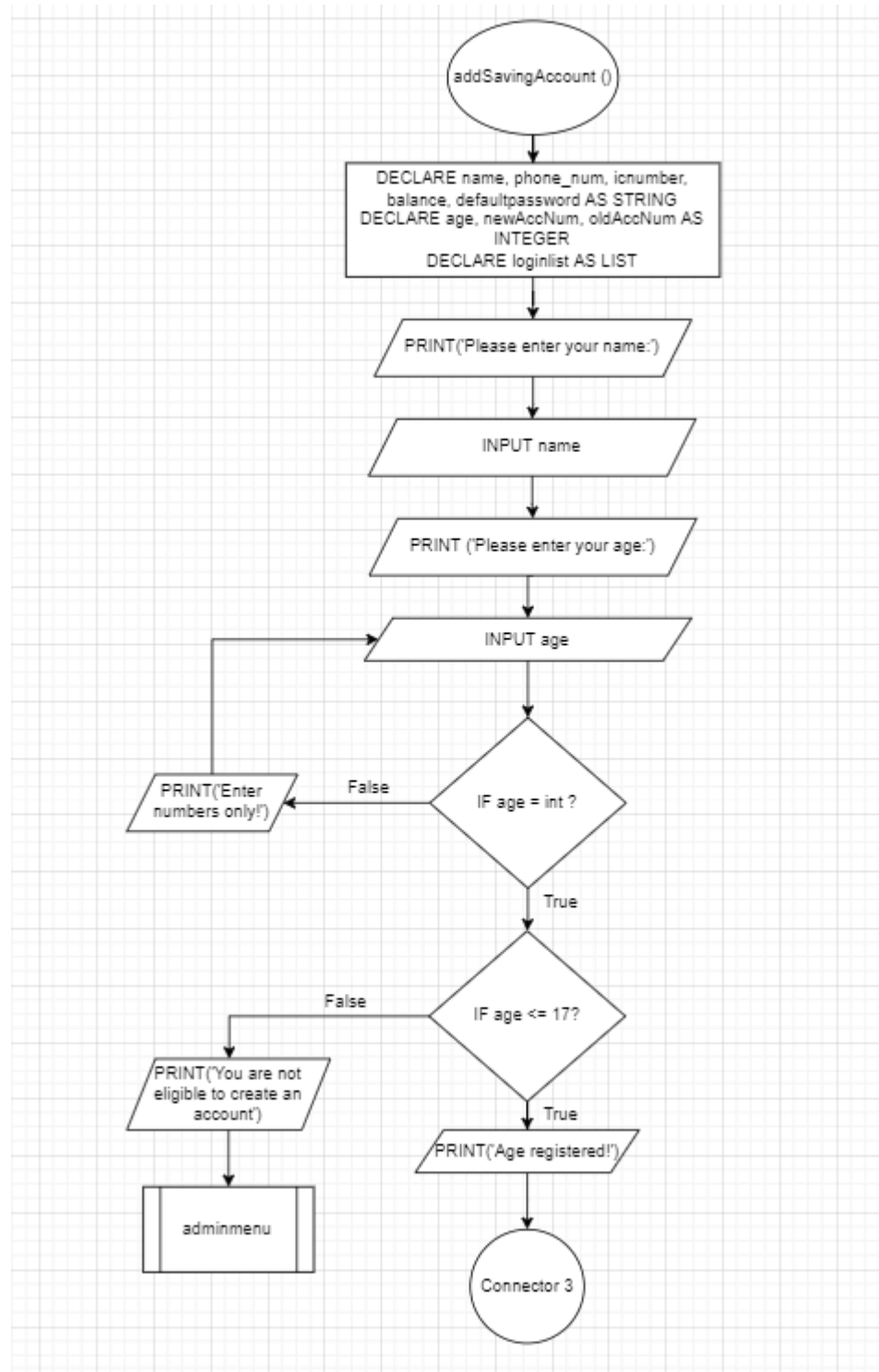
**Login customer**

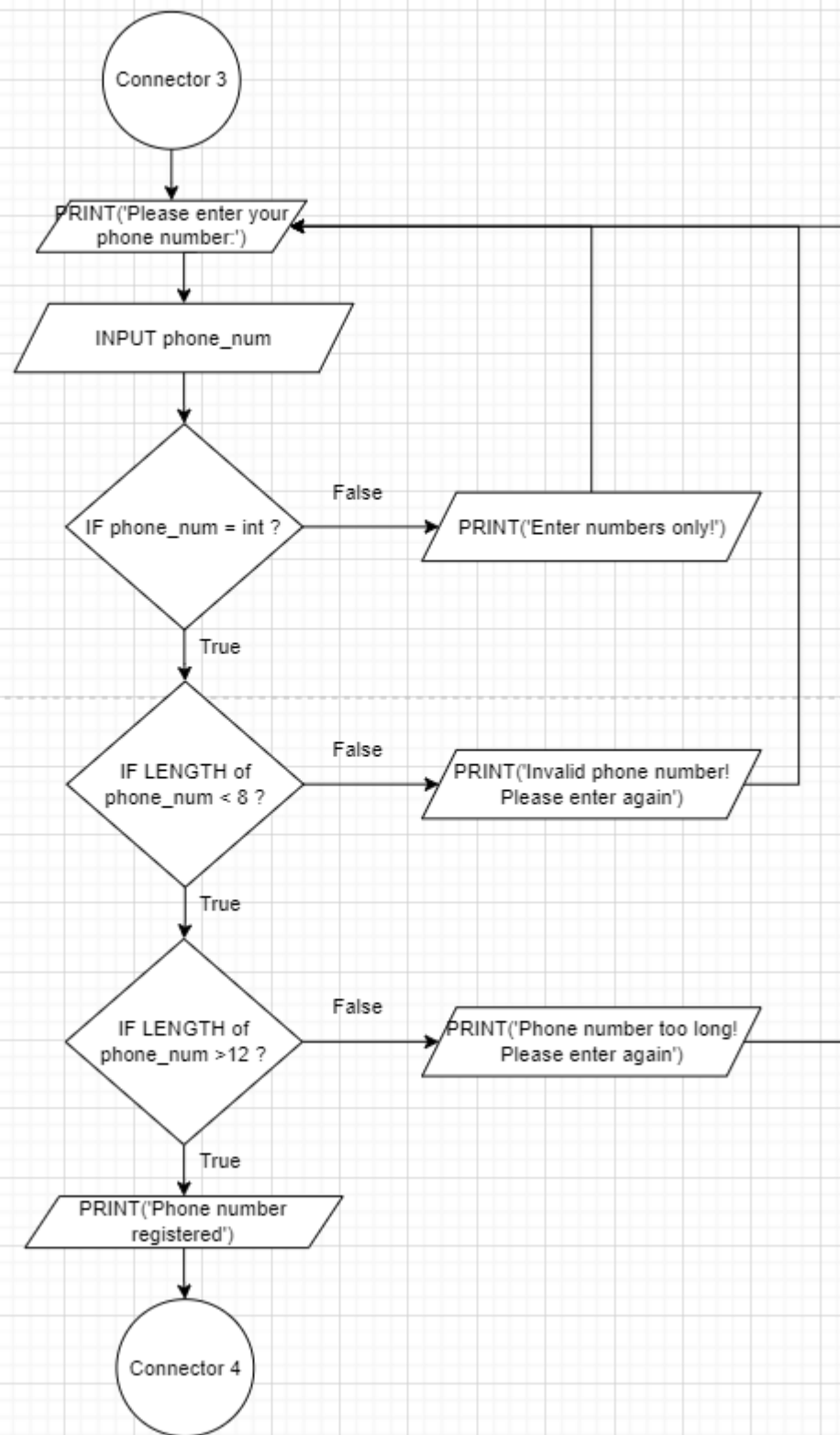
**Admin registration**

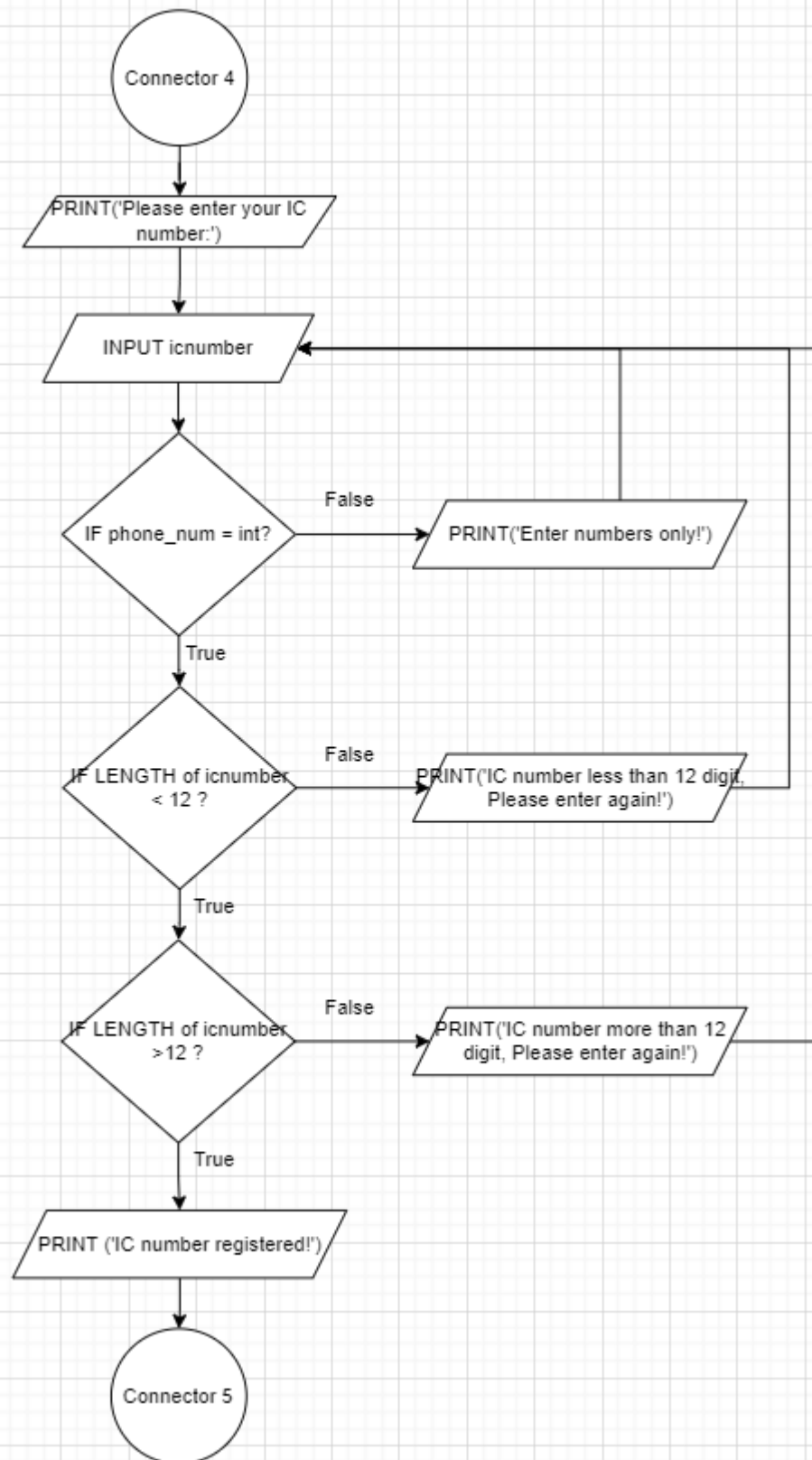


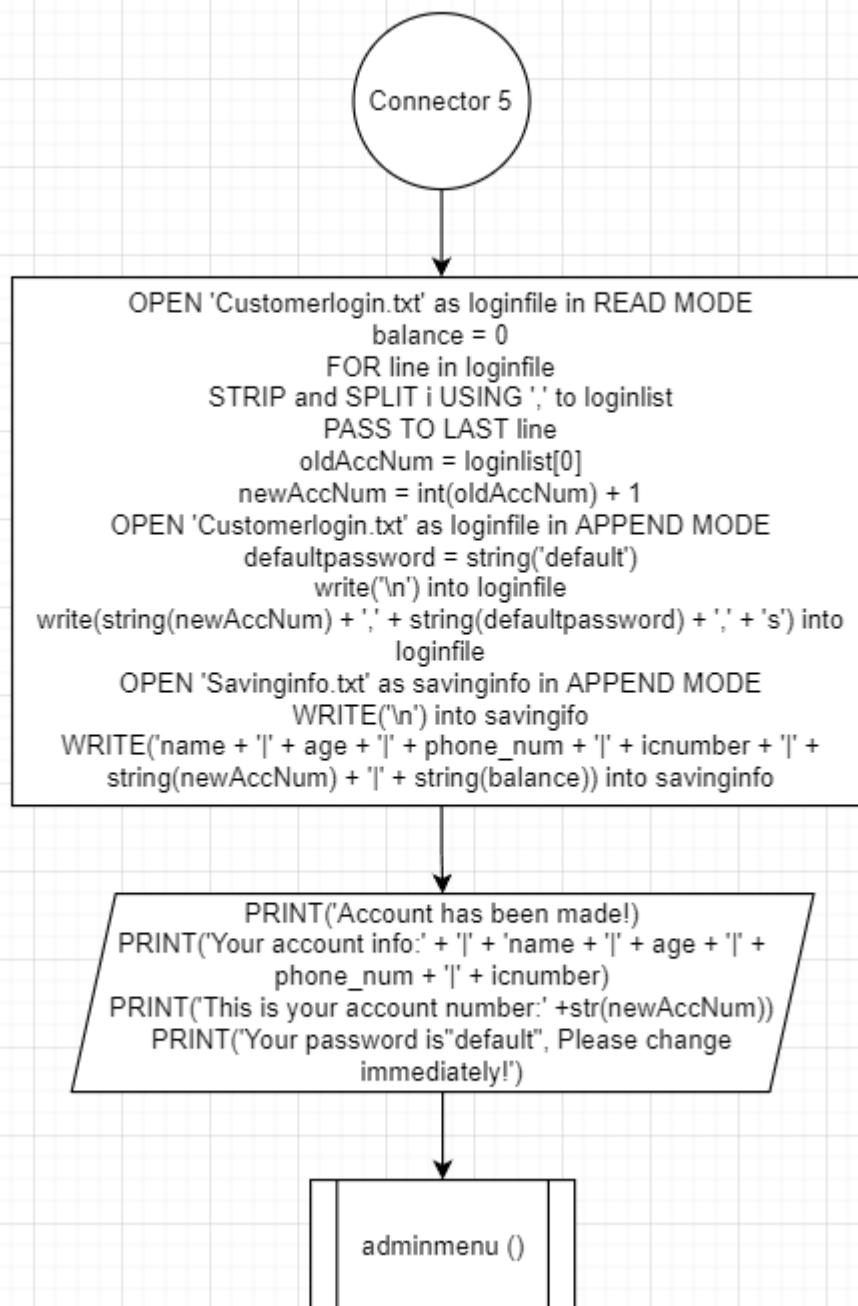


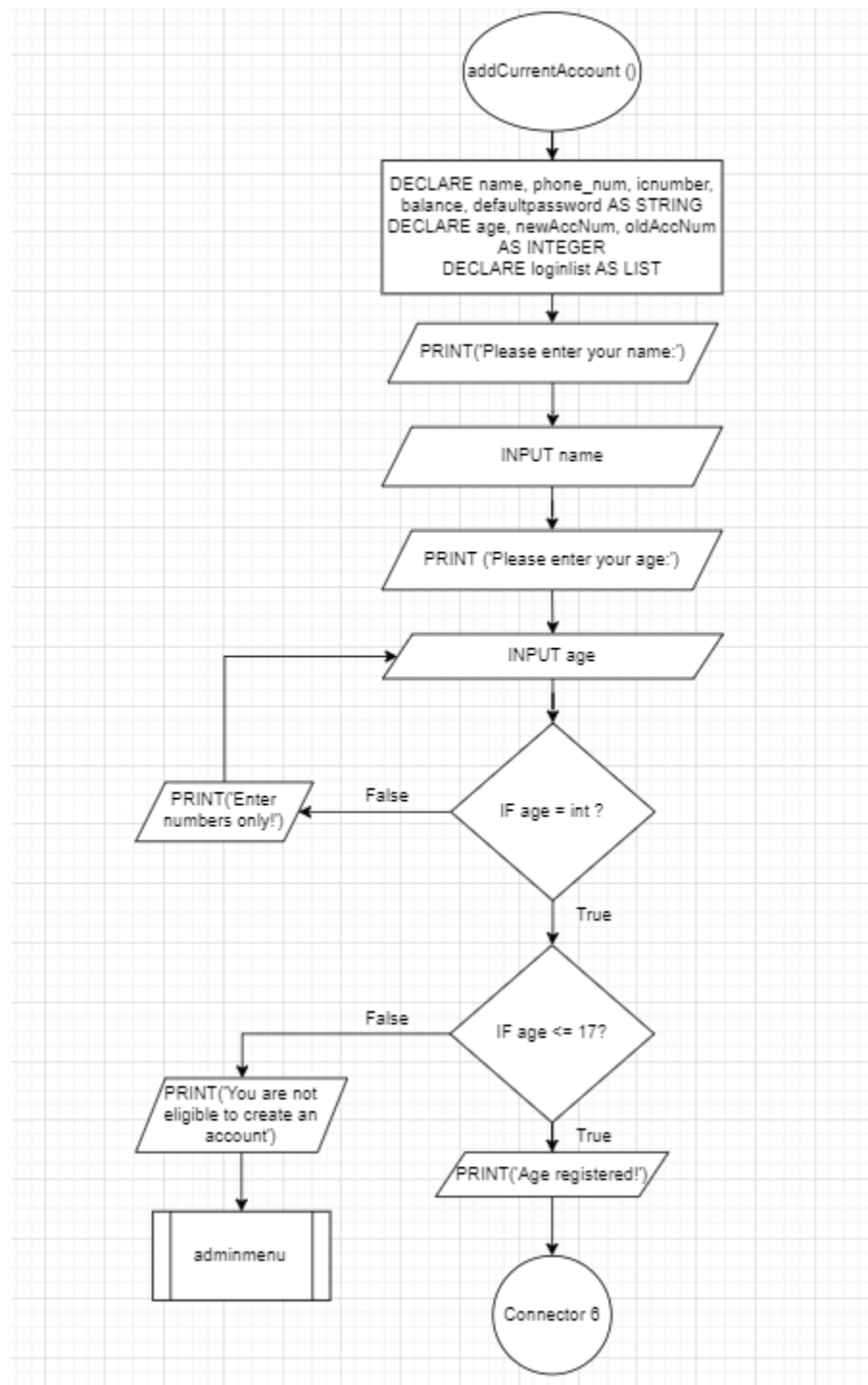
**Change customer password**

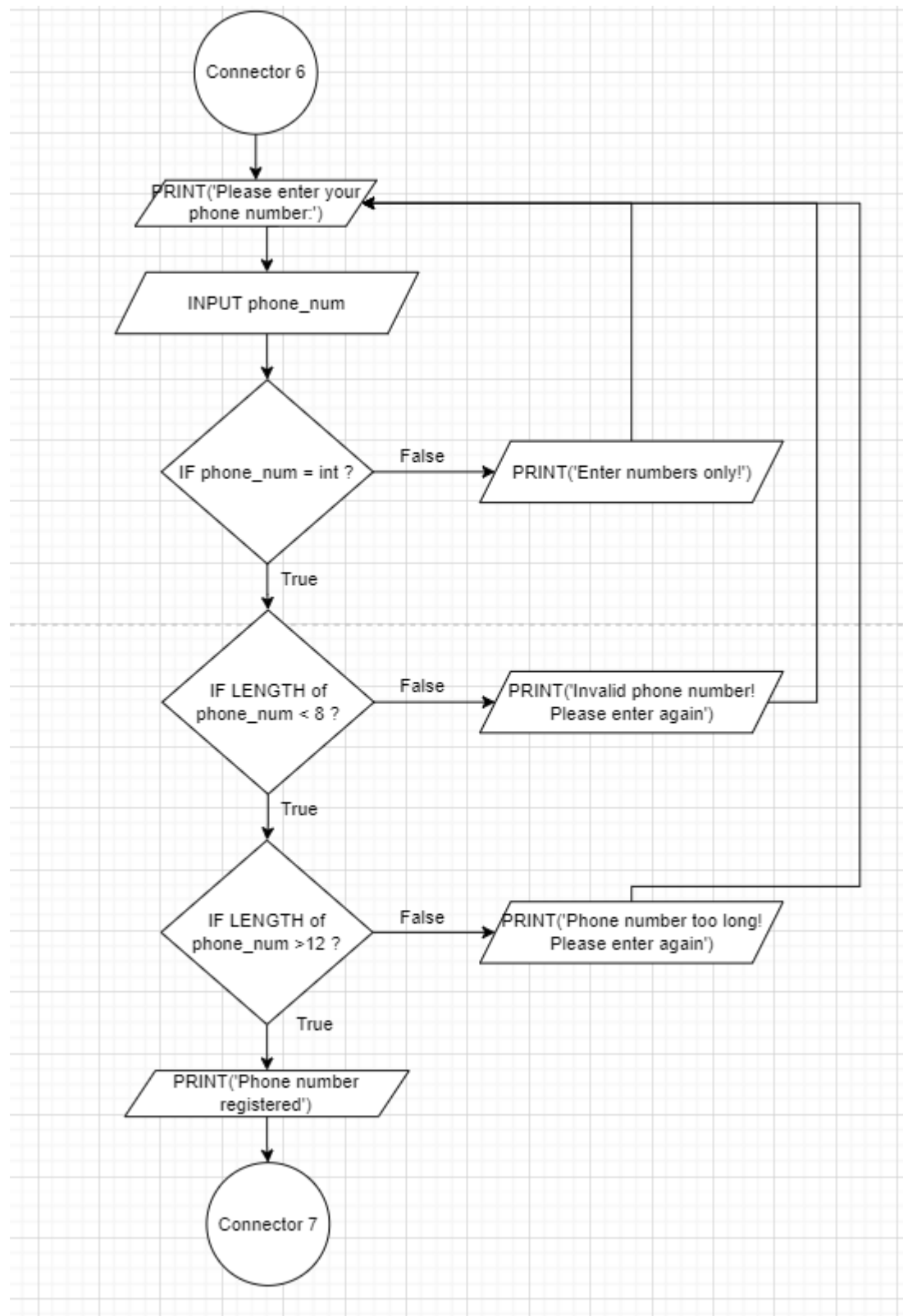
**Saving customer account registration**



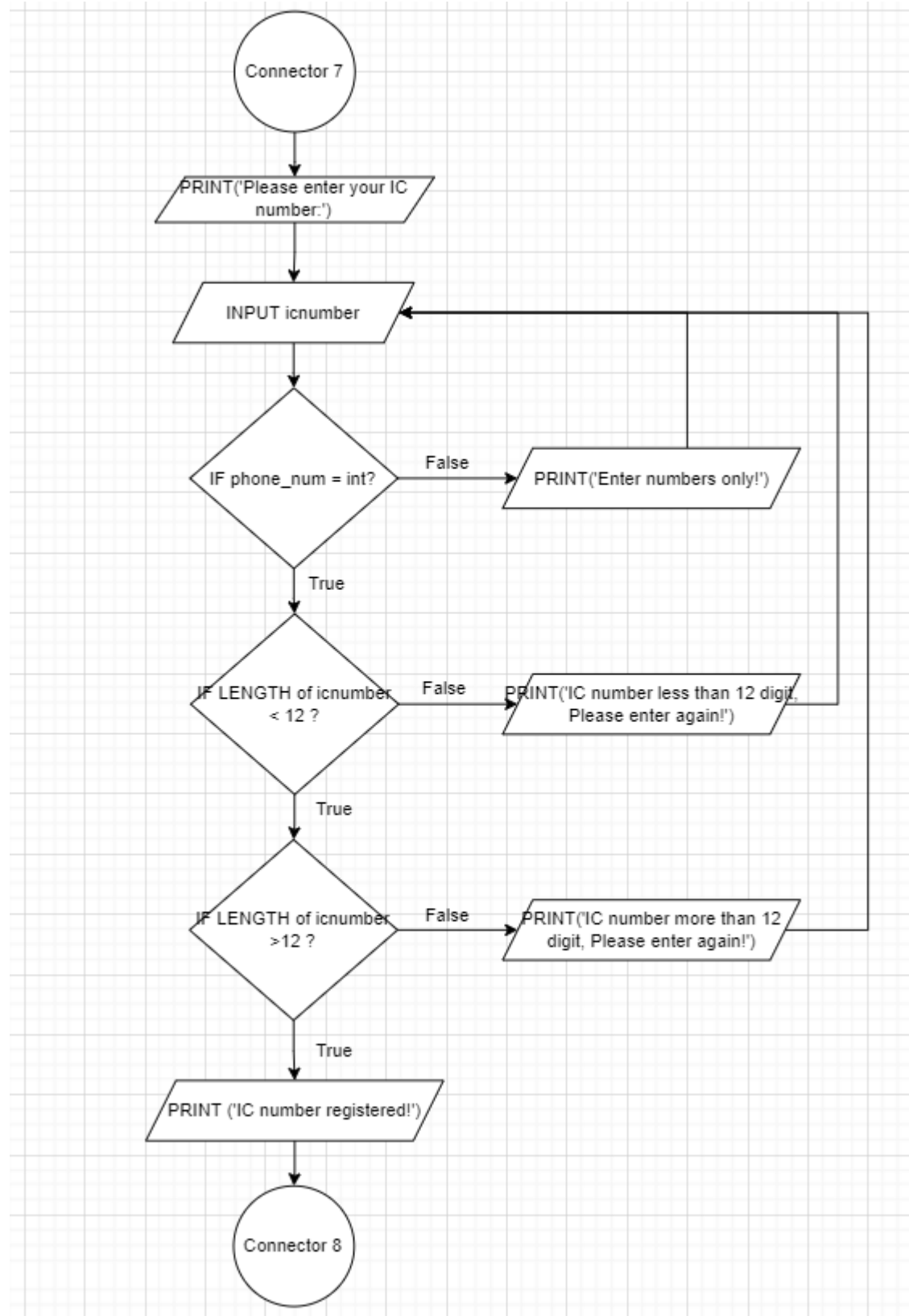


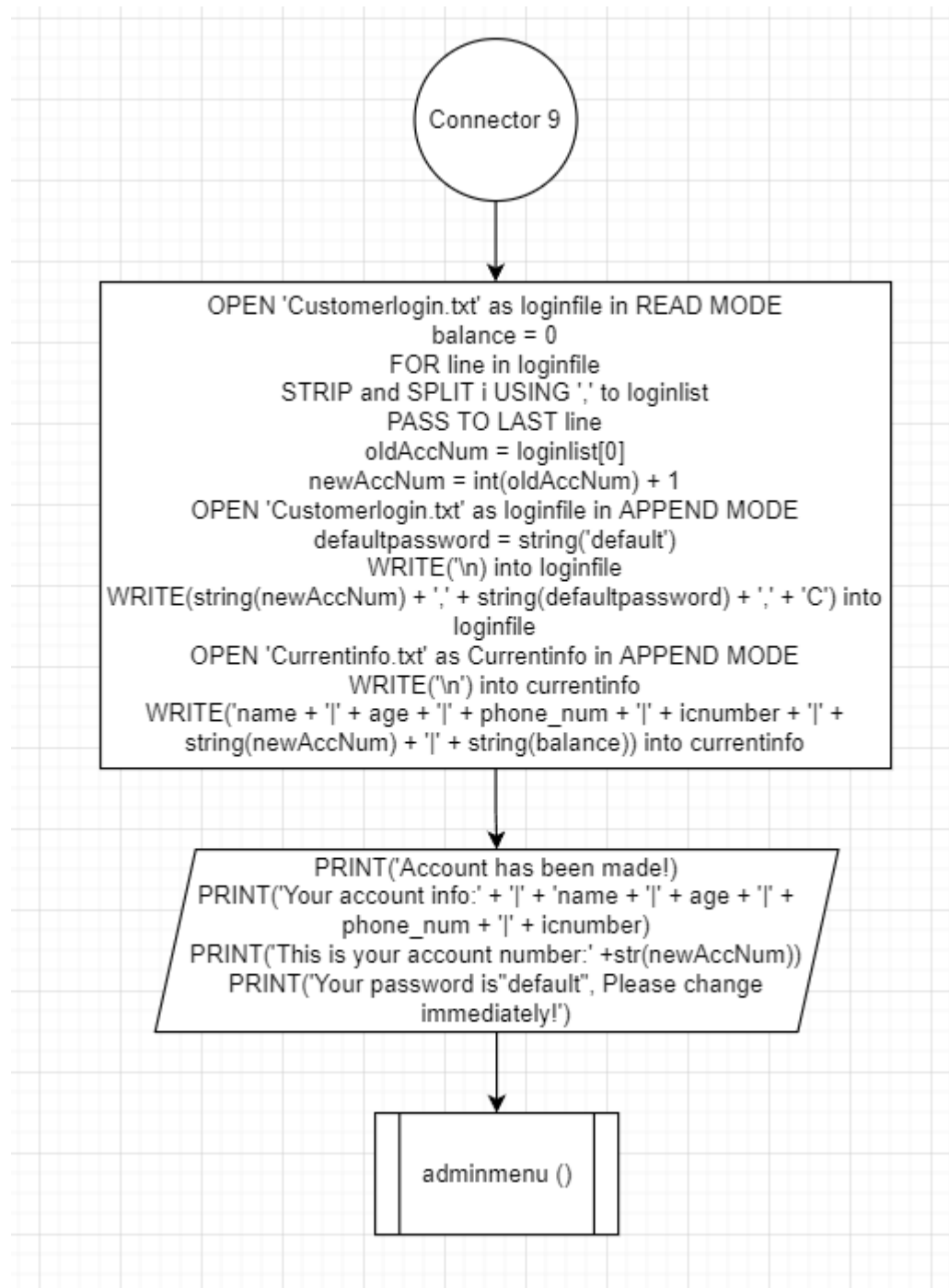


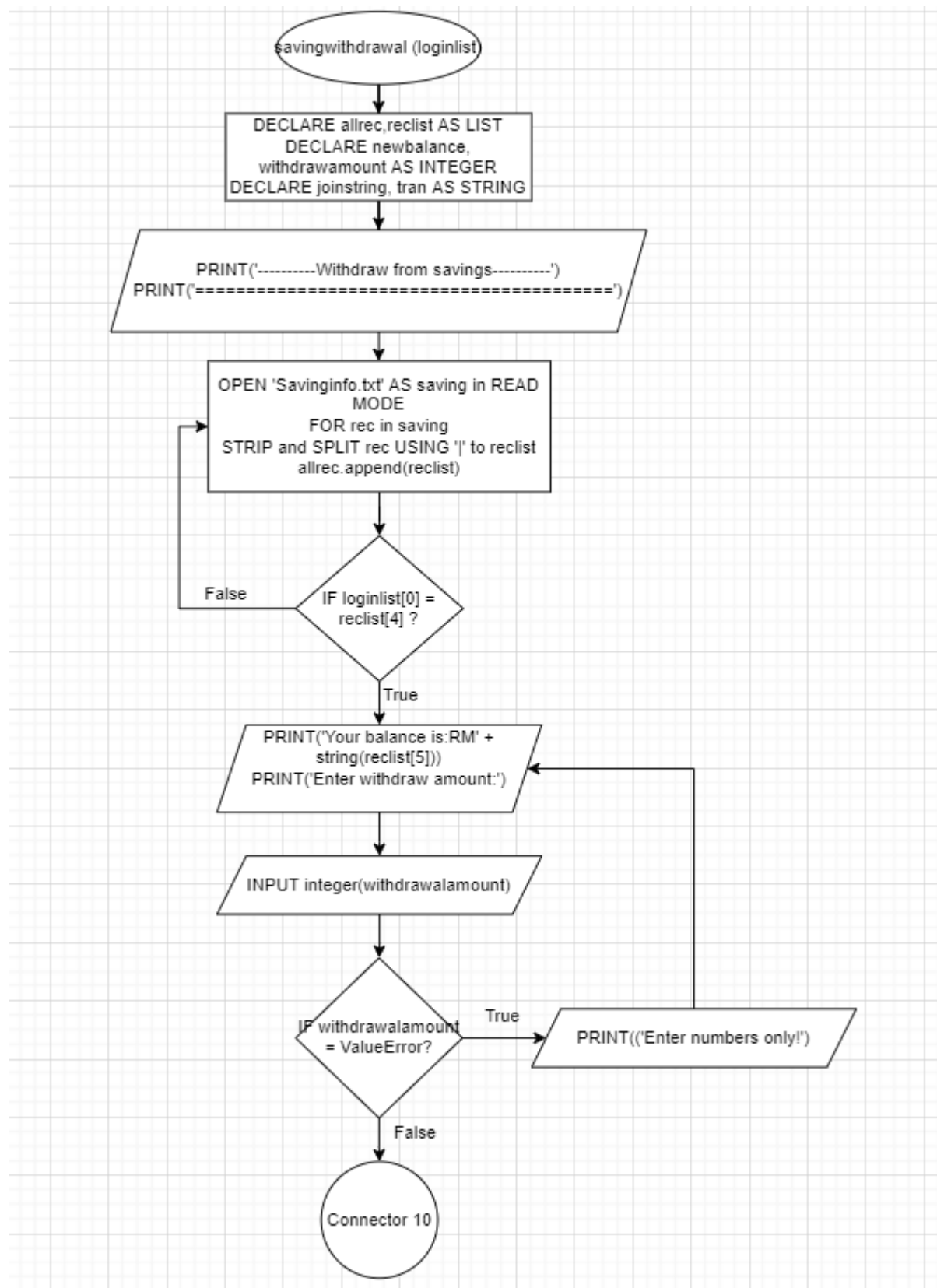
**Current account registration**

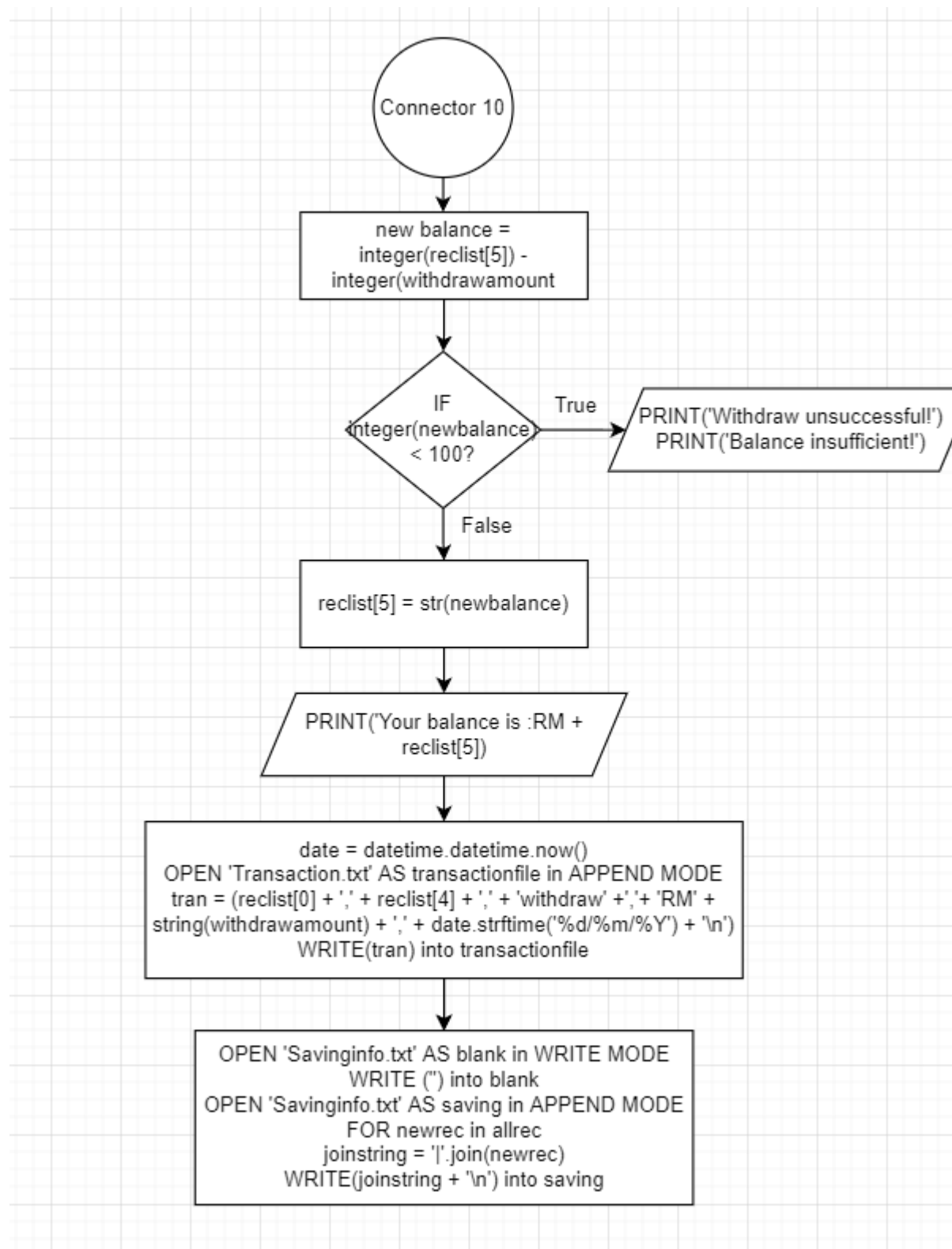


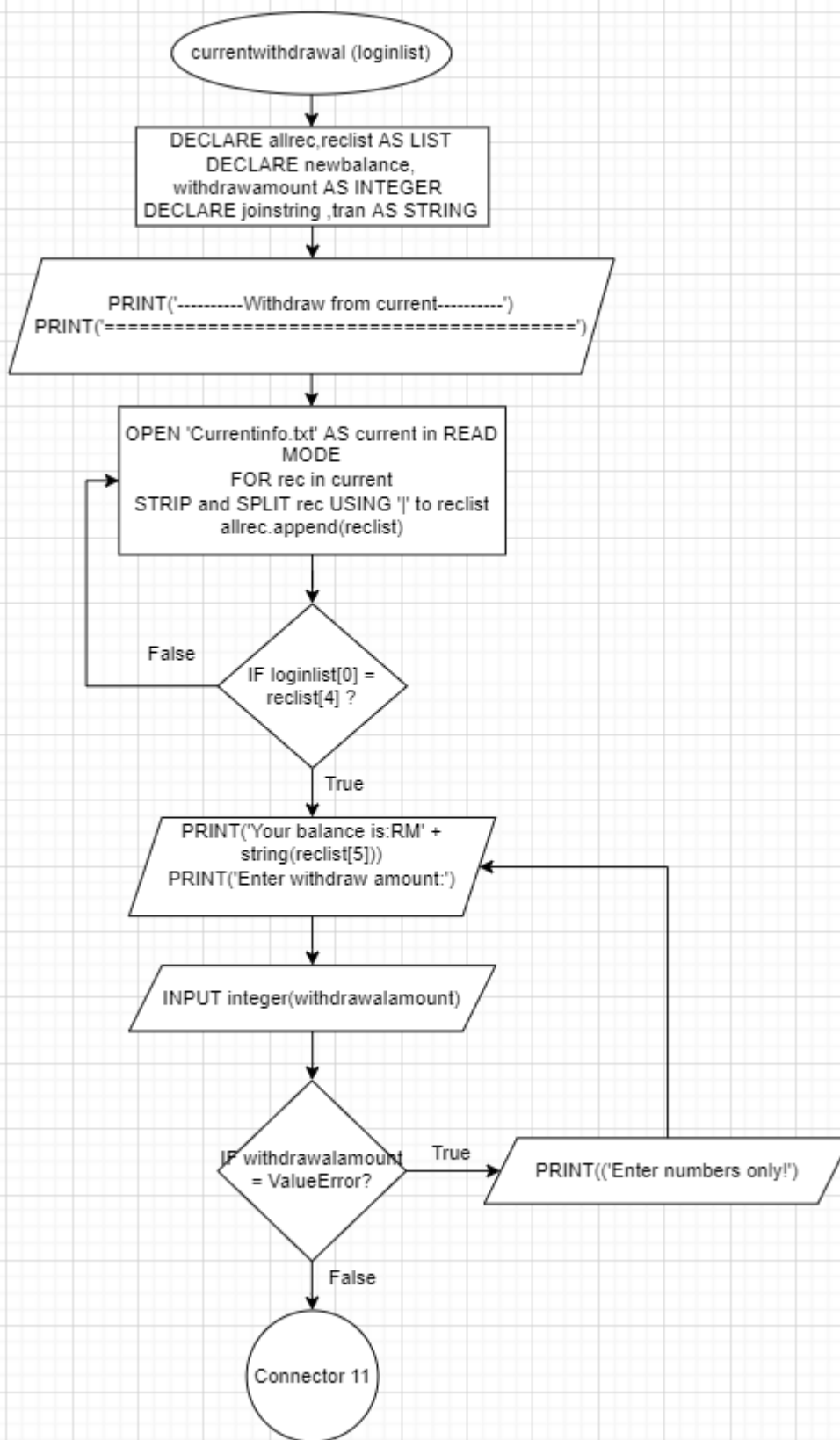


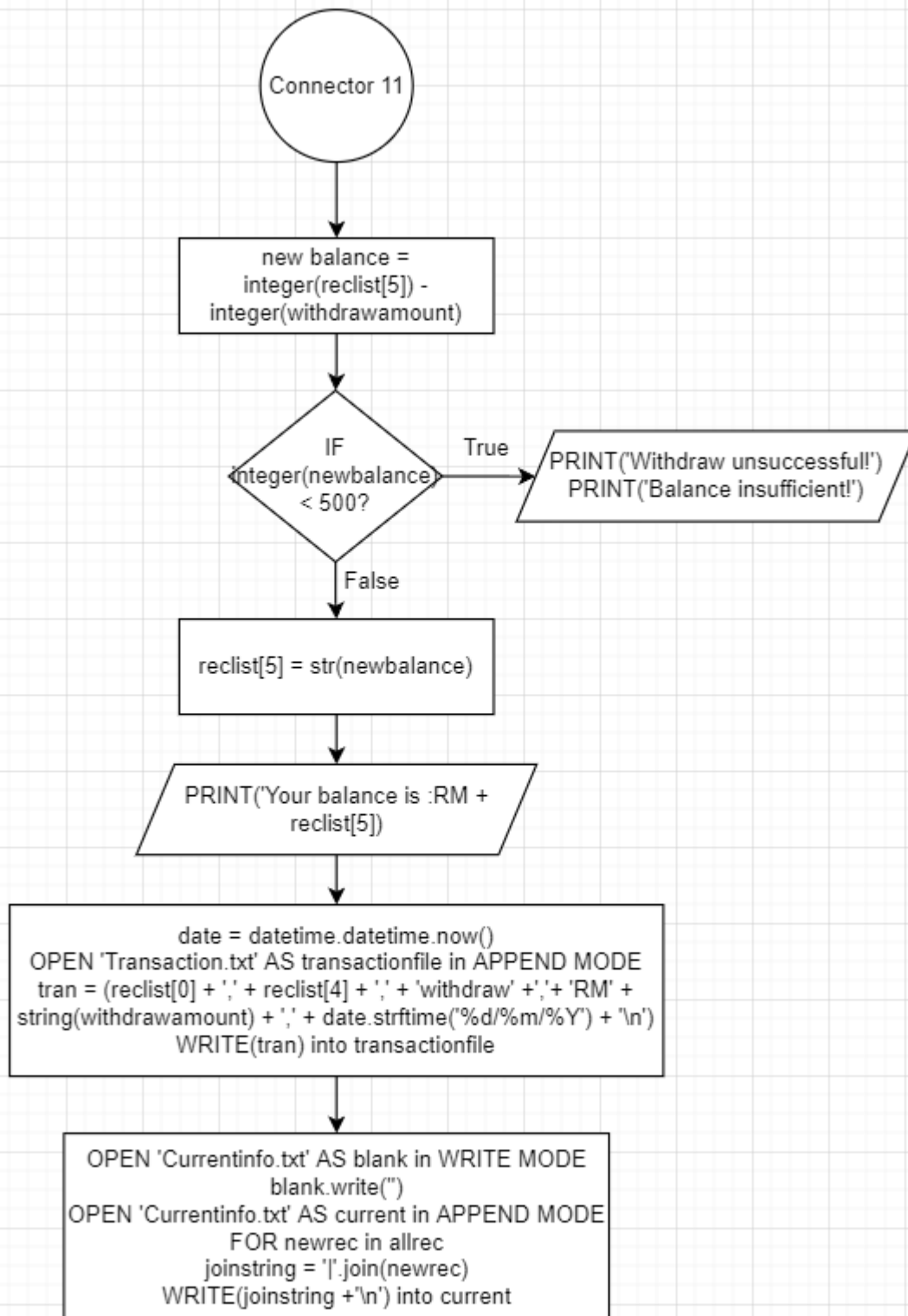


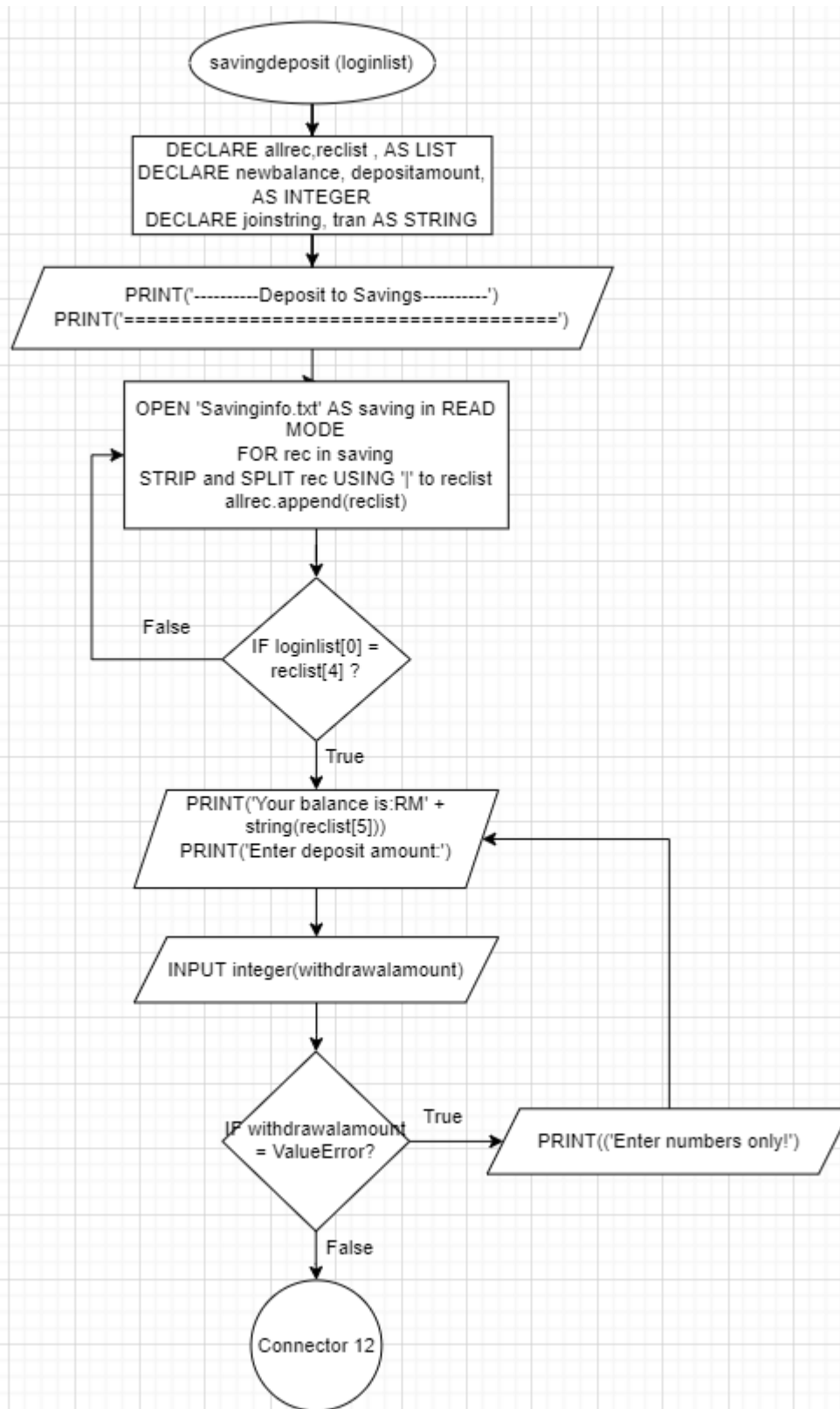


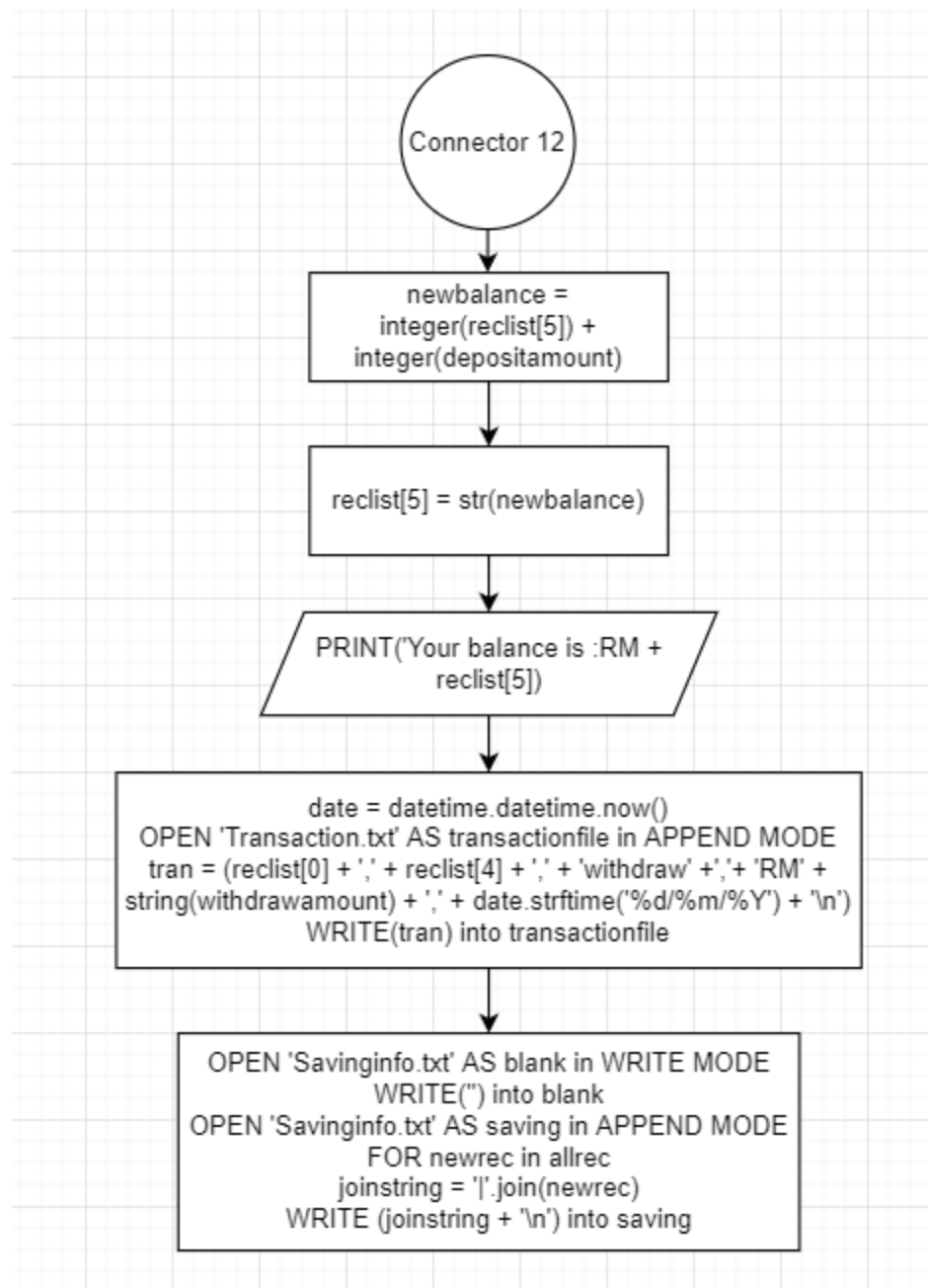
**Saving customer withdrawal**



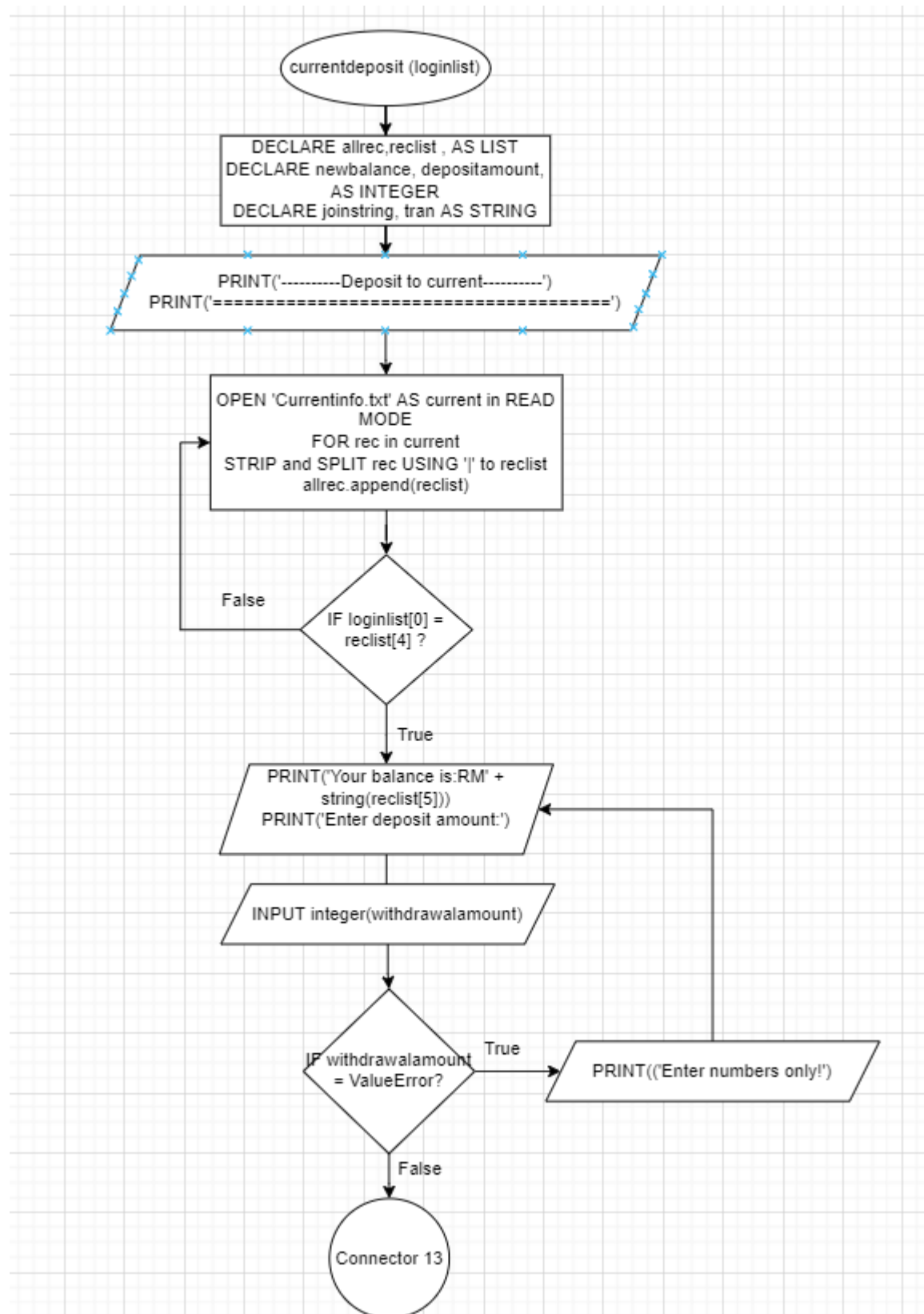
**Current customer withdrawl**

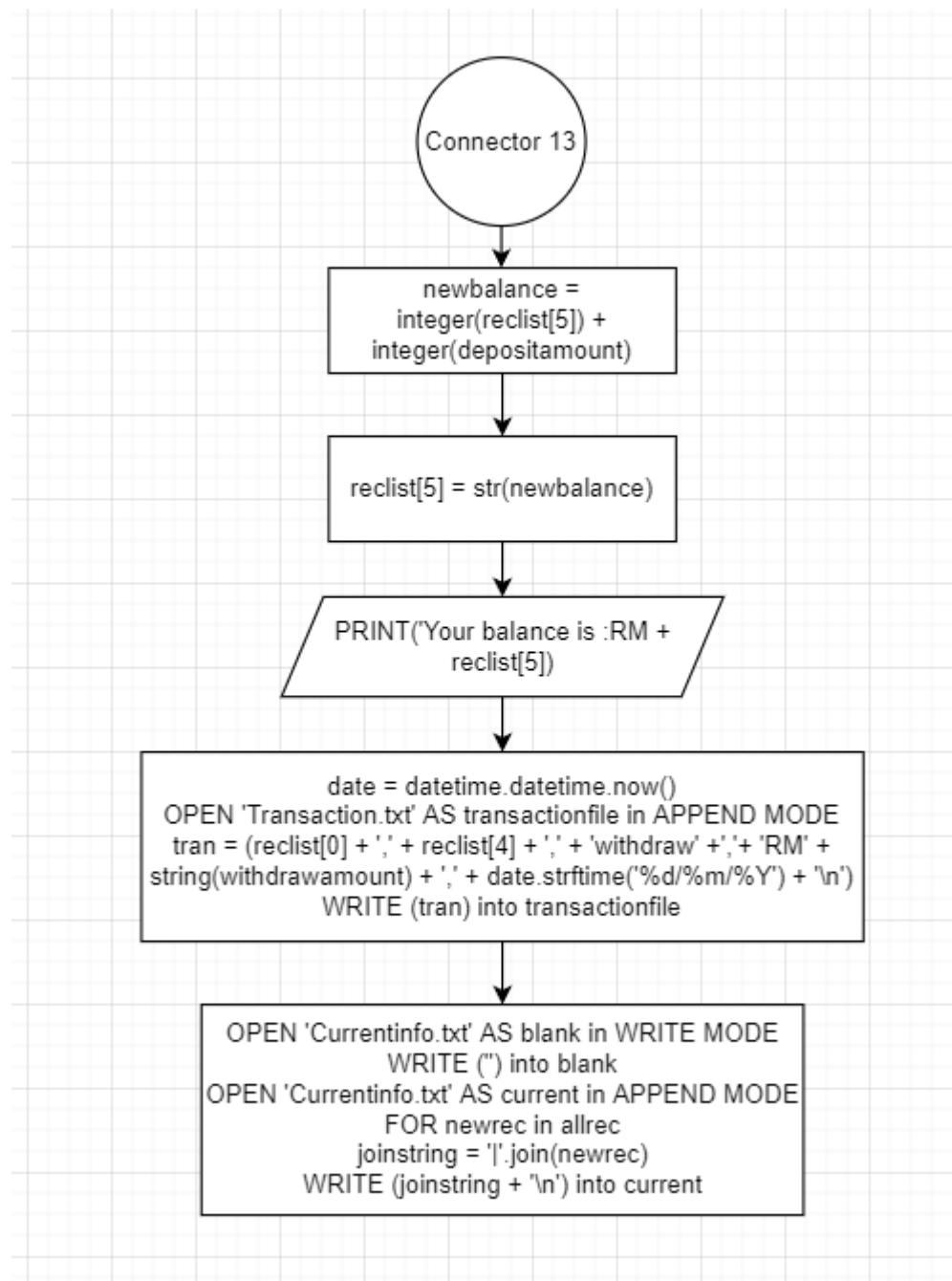


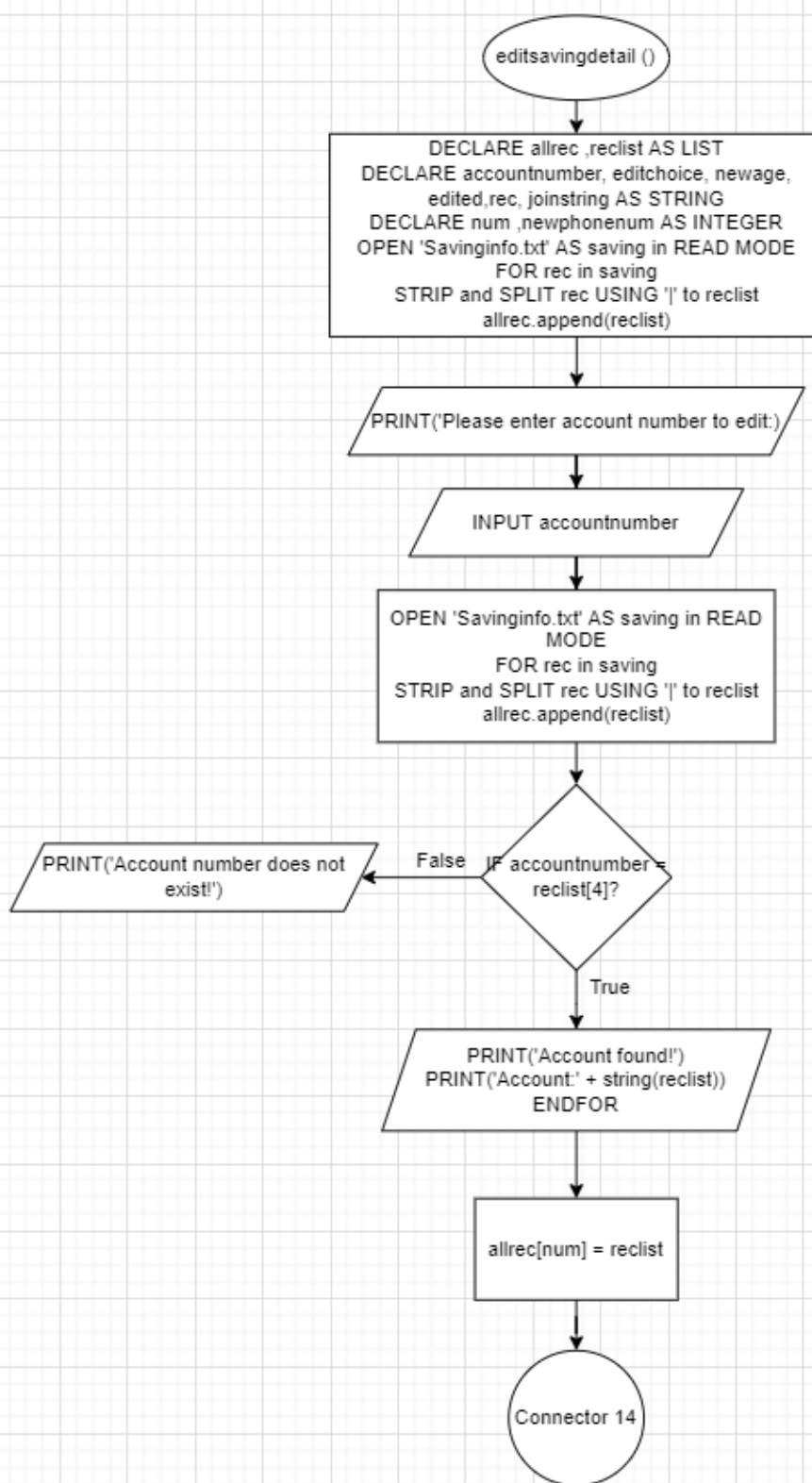
**Saving customer deposit**

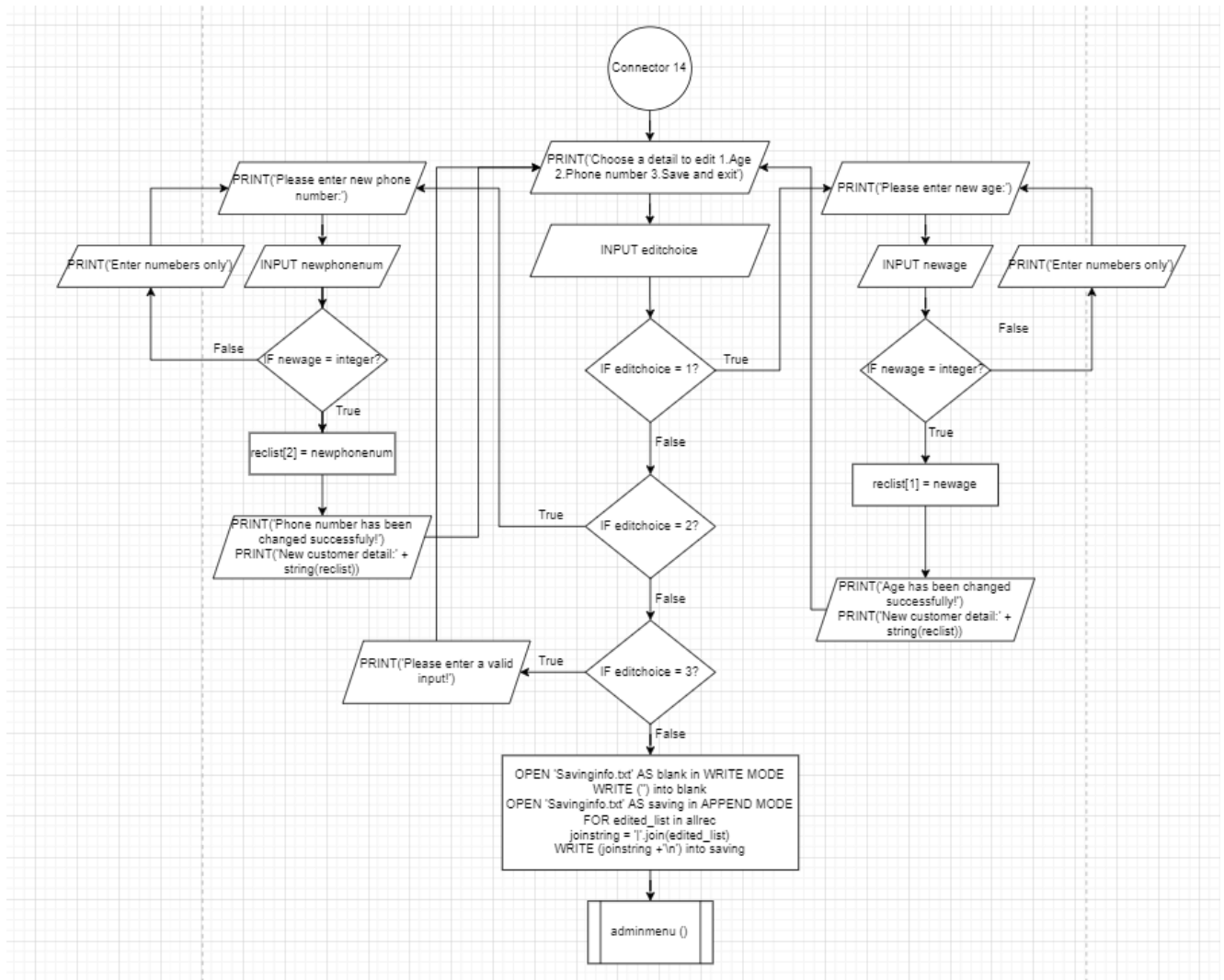


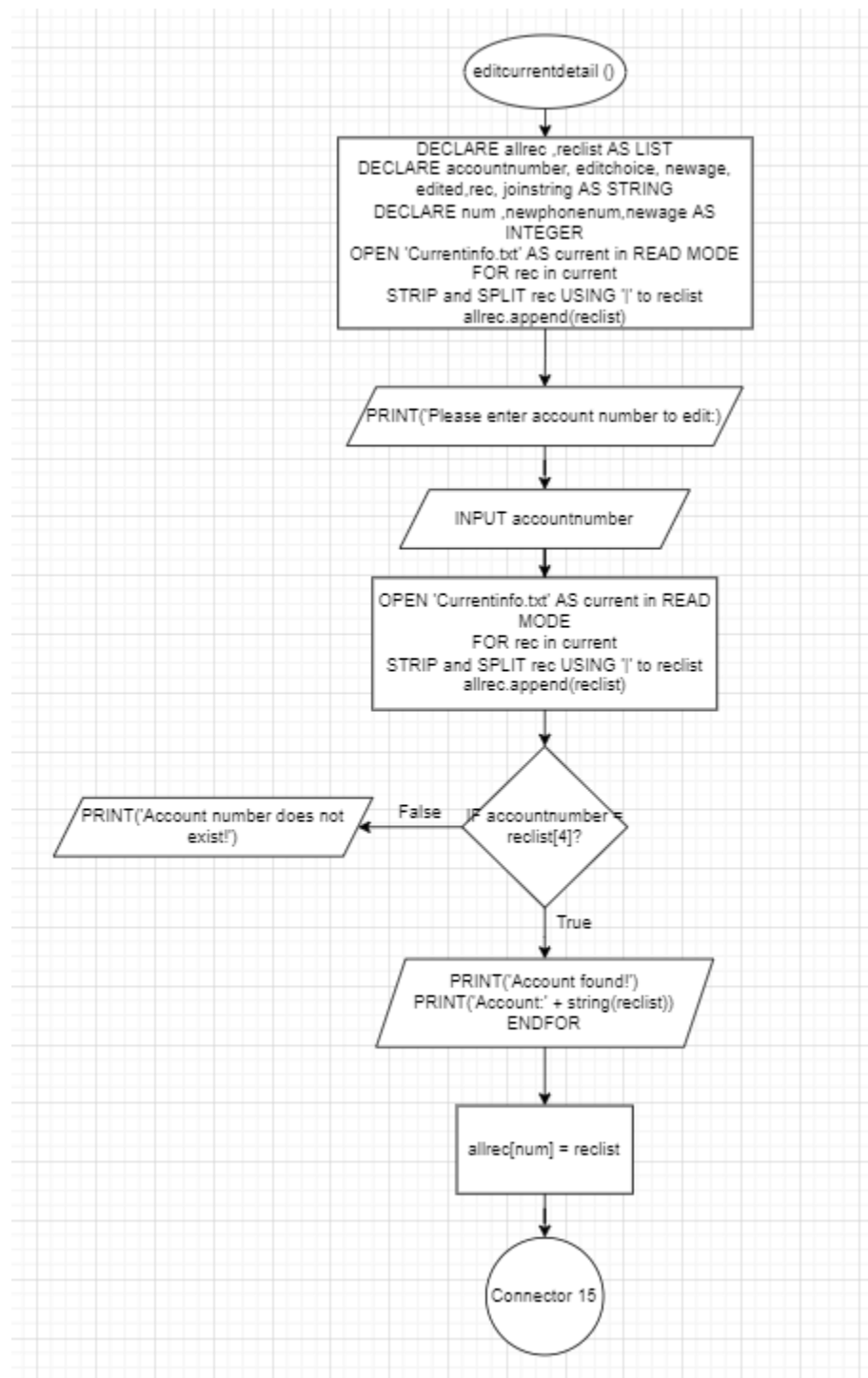


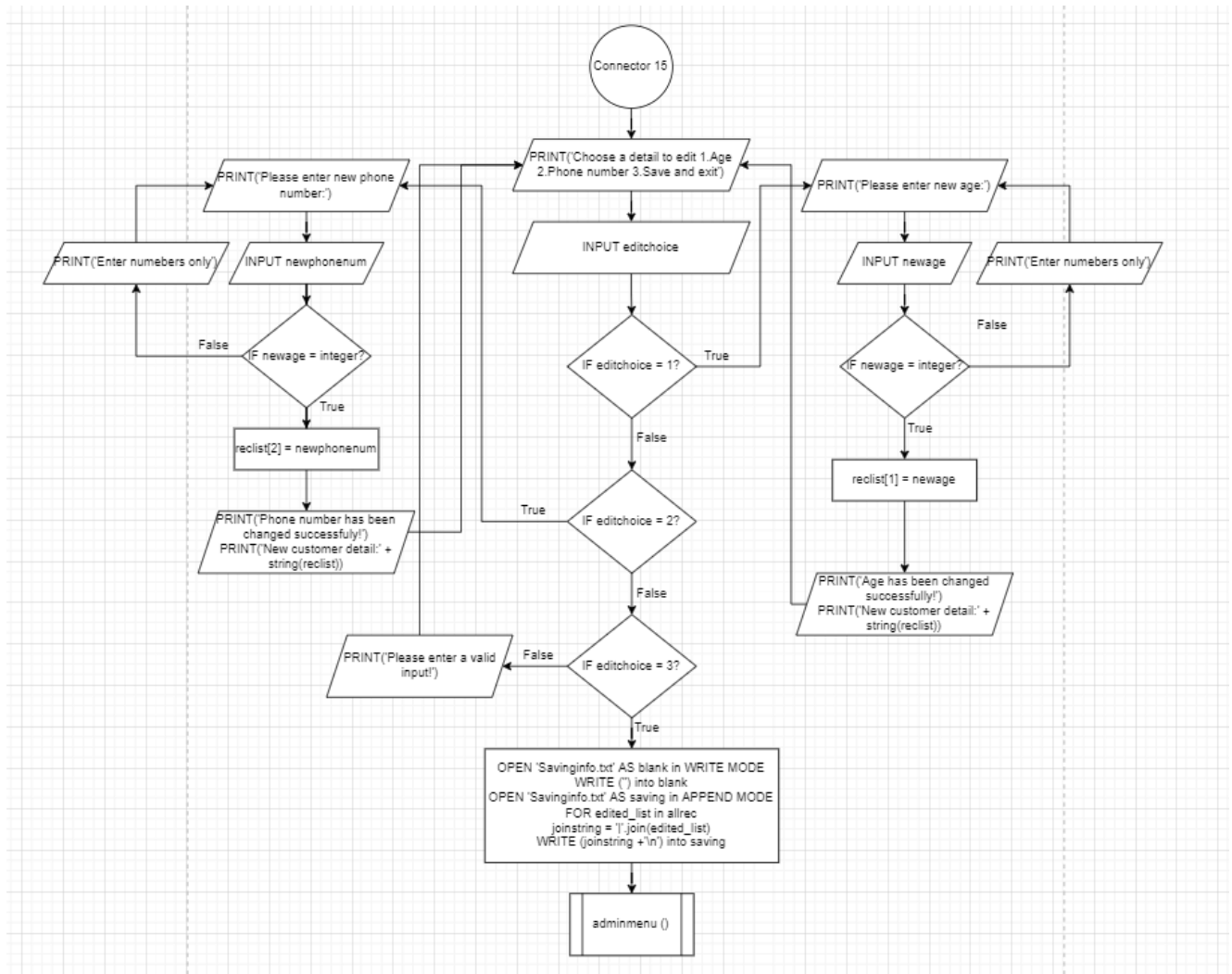
**Current customer deposit**

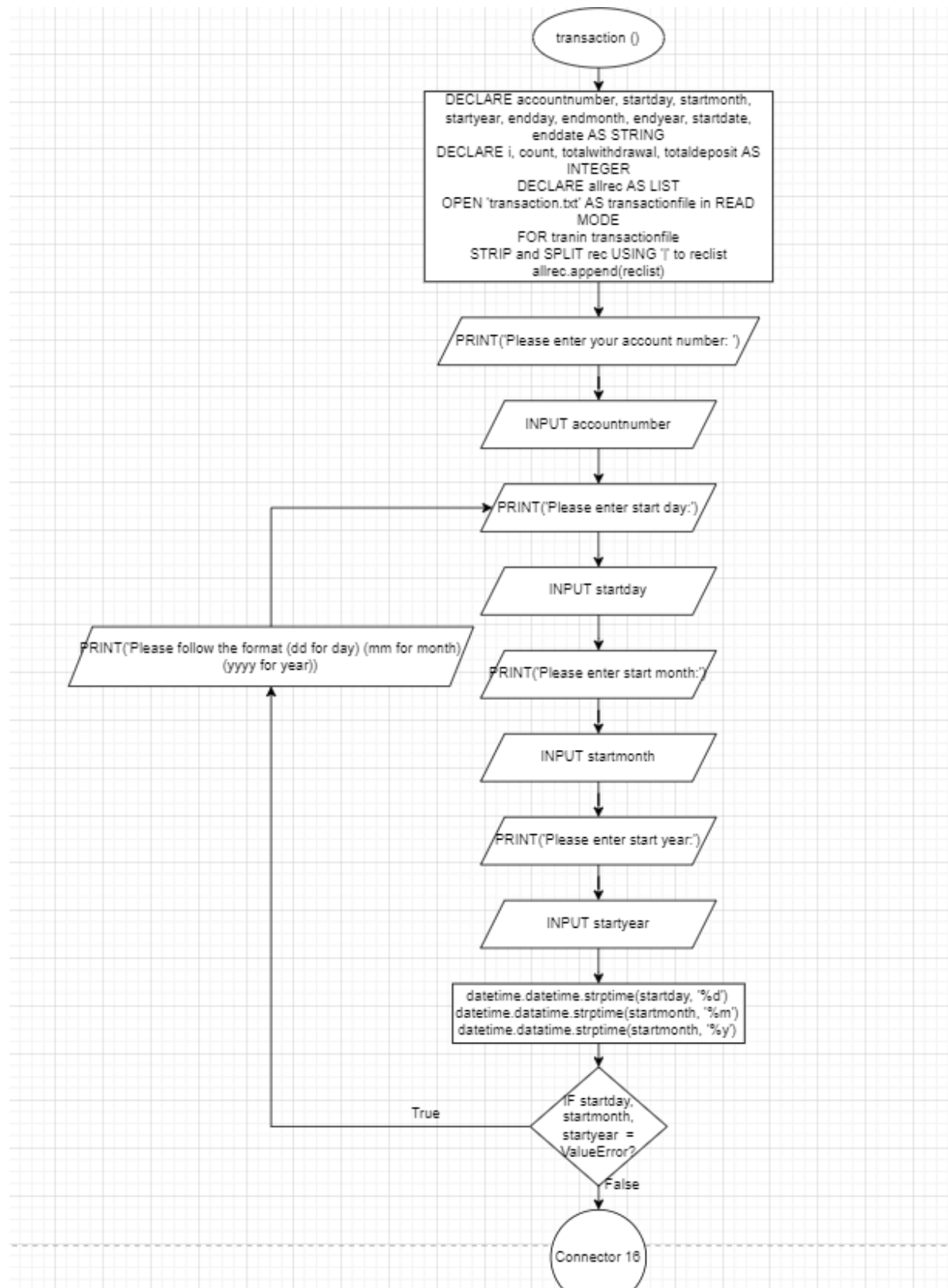


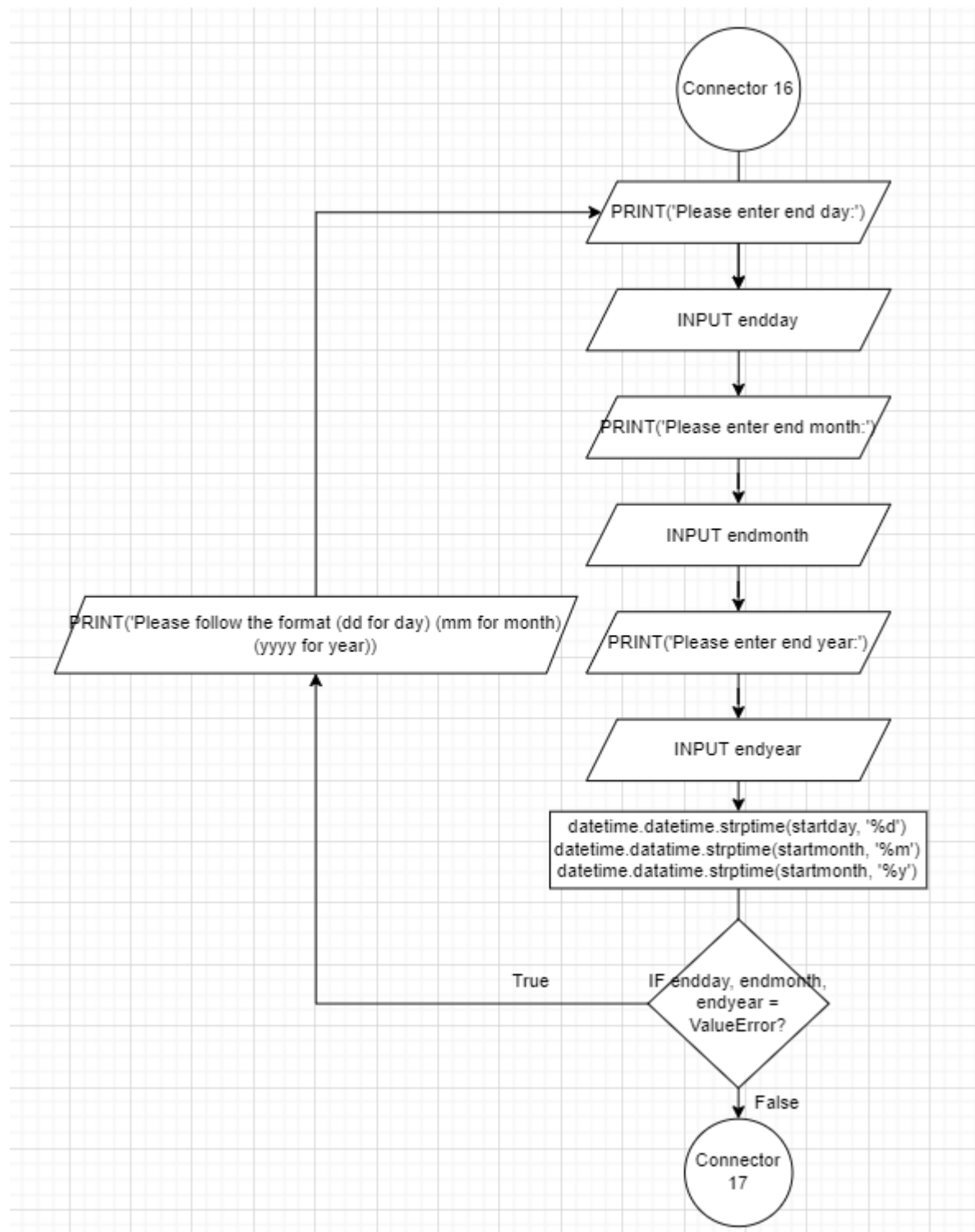
**Edit saving customer's detail**



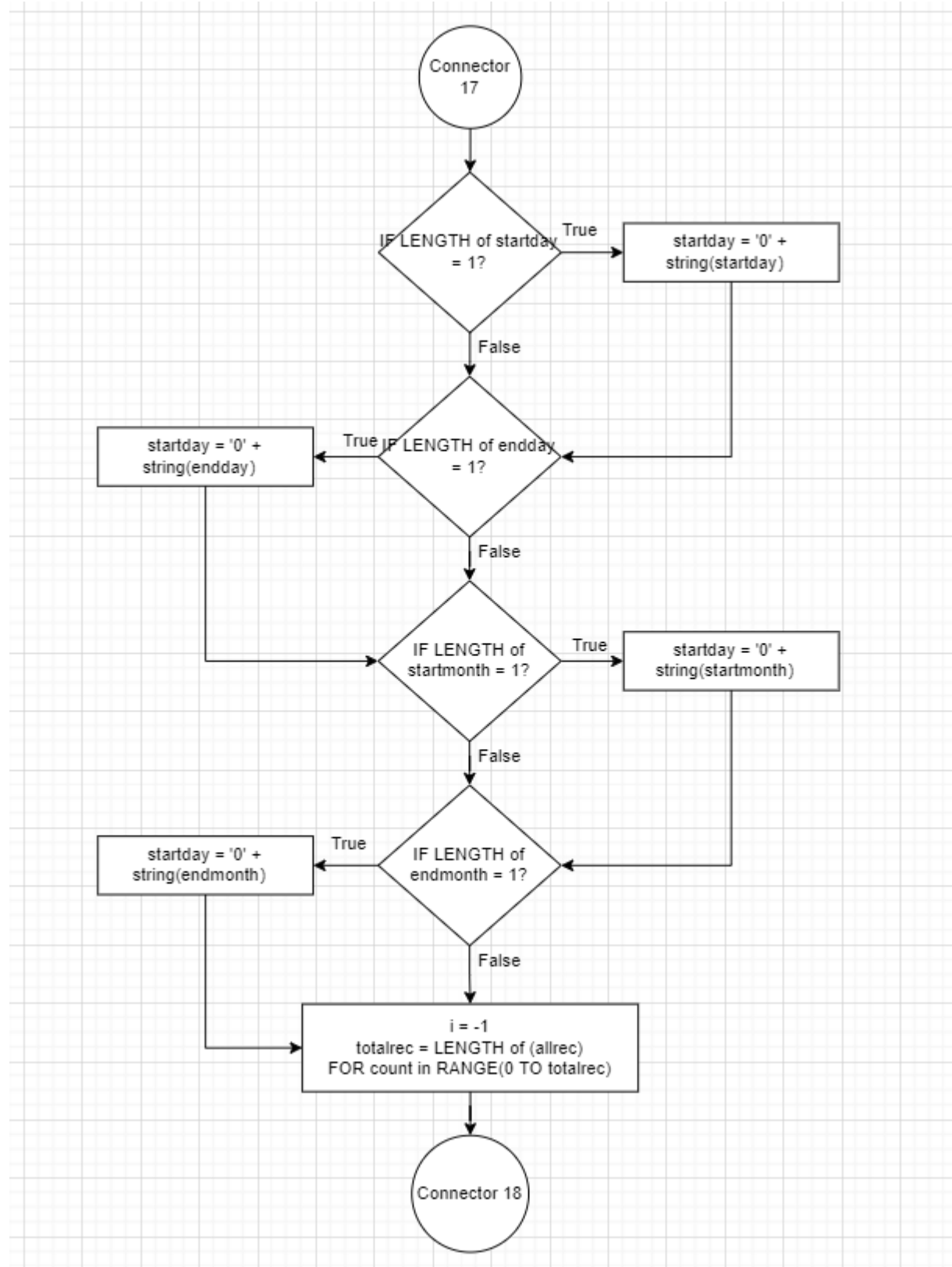
**Edit current customer's detail**

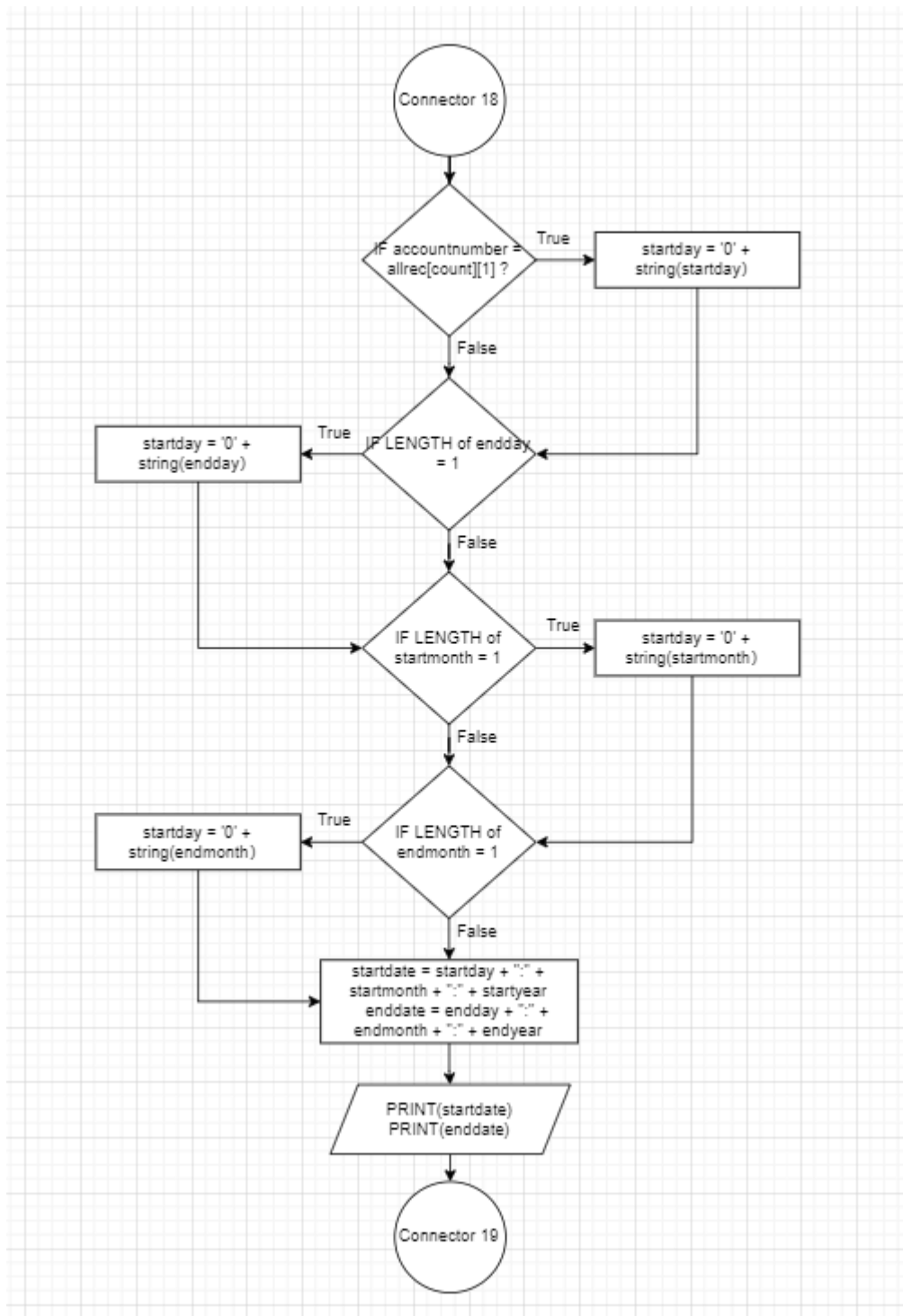


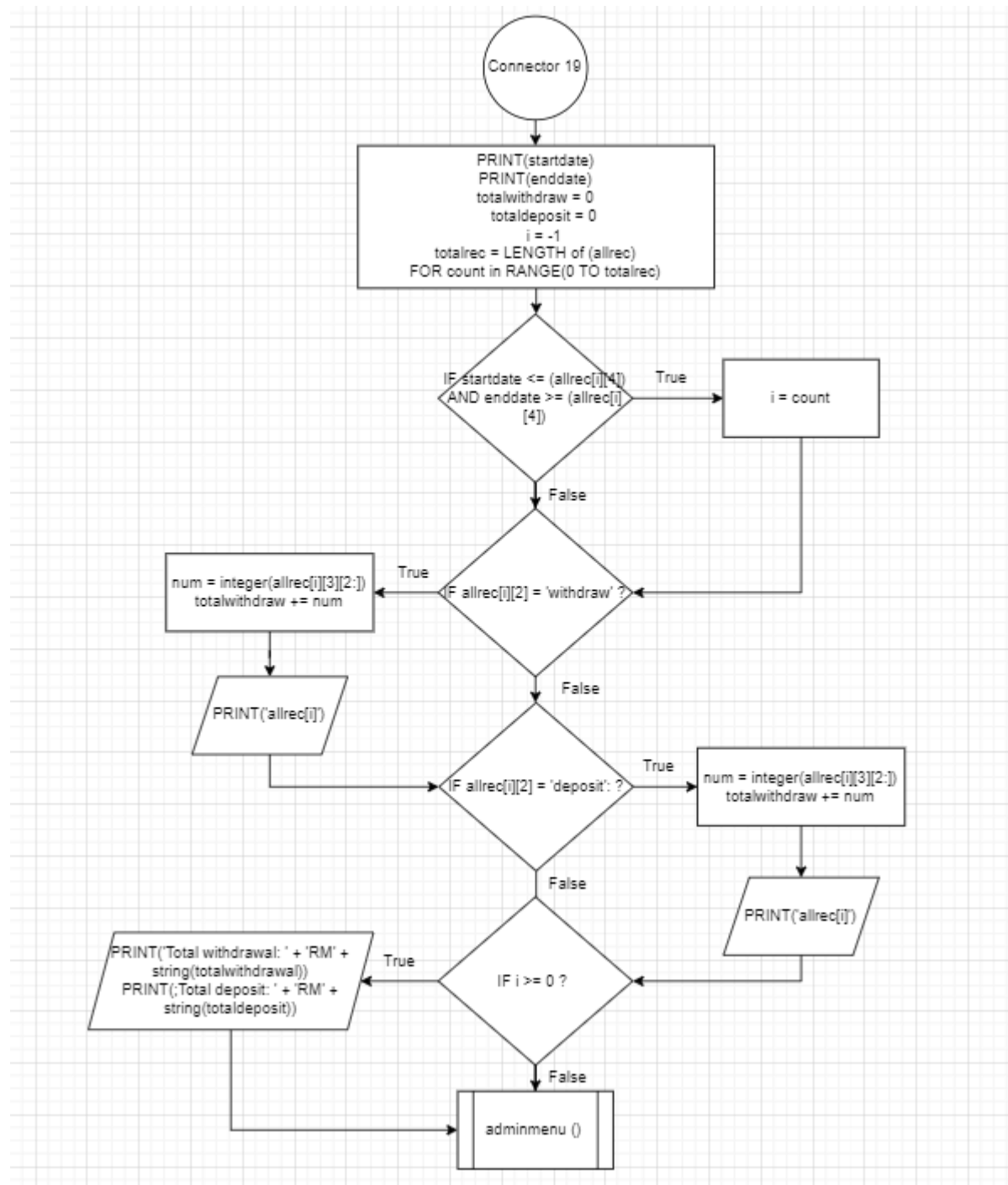
**View customer transaction**











## **Programming Concepts with source code for explanation**

Program source code is assembly code that is created by humans and easy to understand by people as it is similar to English language format but not binary format. However, computers cannot understand such programming languages. So, language translators will take the responsibility to convert the source code into object code that computers can understand (GeeksforGeeks, 2020).

### **1.0 Variable**

A variable act as a reserved memory location that can be stored with different value. It can be string, integer, Boolean and list. In this program, the variable names are given according to what it is supposed to store. Variables' value can be decided by user's input or by the programmers.

#### **1.1 String**

```
defaultpassword = str('default')
```

*Figure 1.1*

As shown in figure 1.0 above, a variable named 'defaultpassword' is given a string value of 'default'. By doing this, it will be easier when programmer needs to store the value in a file.

#### **1.2 Integer**

```
age = int(input('Please enter your age\n: '))
```

*Figure 1.2*

In figure 1.2 above, the value of the variable named 'age' is given by the user's input while displaying 'Please enter your age'. After the user input their age, the value of the input will be having integer value.

#### **1.3 Boolean**

```
success = False
```

*Figure 1.3.1*

```
success = True
```

*Figure 1.3.2*

Boolean's value can only be either True or False. Figures 1.3.1 and 1.3.2 has shown that the variable 'success' have only 2 values that act as a criteria for the program to run 2 different things.

#### 1.4 List

```
allrecord = []
```

*Figure 1.4.1*

```
reclist = rec.strip().split(',')
```

*Figure 1.4.2*

According to figure 1.4.1 and figure 1.4.2, these both variables are given the value of list to store data. In 1.4.1, the '[]' means that the variable 'allrecord' is an empty list. While in 1.3.2, variable 'reclist' is a list that made from splitting the variable 'rec'.

#### List length

```
totalrec = len(allrec)
```

*Figure 1.4.3*

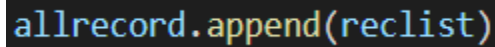
The figure 1.4.3 use 'len' to get the length of value of 'allrec' and enter to the variable 'totalrec'

#### List elements

```
reclist[5] = str(newbalance)
```

*Figure 10.4.4*

Figure 10.4.4 shows how a new element of a list is entered. In a list, the 1<sup>st</sup> element is positioned at number 0. Therefore, 'newbalance' is entered as a value for the 6<sup>th</sup> element of 'reclist'.

List append


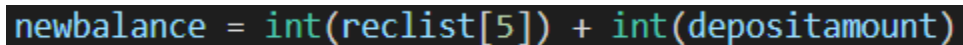
```
allrecord.append(reclist)
```

Figure 10.4.5

‘append’ is a way to add a new element to a list. In figure 10.4.5, it is appending the variable ‘reclist’ into the list named ‘allrecord’. It can append any variable into the list including list that will make the ‘allrecord’ list becomes a nested list.

2.0 Operator

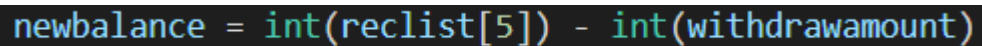
Operators are special symbols on python code that can complete arithmetic or logical computation.

2.1 Addition


```
newbalance = int(reclist[5]) + int(depositamount)
```

Figure 2.1

In figure 2.1 shown above, this is to calculate the balance after the user has deposited to their account. The 6<sup>th</sup> element of the list named ‘reclist’ and ‘depositamount’ are given integer values. It is then summed using ‘+’ symbol. Then the result will become the value of newbalance.

2.2Reduction


```
newbalance = int(reclist[5]) - int(withdrawamount)
```

Figure 2.2

In figure 2.2 shown above, this is to calculate the balance after the user has withdrawn from their accounts. The 6<sup>th</sup> element of the list named ‘reclist’ and ‘depositamount’ are given integer values. It is then summed using ‘-’ symbol to perform reduction. Then the result will become the value of ‘newbalance’.

### 3.0 Comparison Operator

#### 3.1 Equal

```
if adminid == 'admin' and adminpassword == 'admin':
```

*Figure 3.1*

In figure 3.1 above is taken from the function of 'loginadmin'. The equal symbol '==' is used when comparing 2 values. When the input value for 'adminid' and 'adminpassword' is found equal to 'admin', it will grant access to the super user account.

#### 3.2 Not equal

```
if adminpassword != confirmpassword:
```

Figure 3.2.1

#### 3.3 Greater than/ Greater than or equal to

```
elif int(newbalance) >= 500:
```

*Figure 3.3.1*

```
elif len(phone_num) > 12:
```

*Figure 3.3.2*

In figures 3.2.1 and 3.2.2 shown above, '>' and '>=' is used differently. In figure 3.2.1, it can be defined as if the value of 'newbalance' more than or equal to, it will run a specific code. In figure 3.2.2, it is used to measure if the length of the variable 'phone\_num' is more than 12.

### 3.4 Less than / Less than or equal to

```
if int(newbalance) < 500:
```

*Figure 3.4.1*

```
if len(icnumber) < 12:
```

*Figure 3.4.2*

In figure 3.3.1 and 3.3.2 above, '<' are used. It can be used with IF to create a condition. For example, it can compare the integer value and also can be compared with the length of a value.

### 4.0 Logical operator

#### 4.1 And

```
if (loginlist[0] == accountnumber and loginlist[1] == customerpassword):
```

*Figure 4.1*

'and' is used to satisfy both criteria to proceed. In this figure 4.1, only the variable accountnumber and customerpassword is equal to the existing list's element will proceed.

### 5.0 Membership operator

#### In

```
for i in adminaccount:
```

*Figure 5.1*

In the figure 5.1 above has shown that 'in' is used with 'for' loop. The variable 'adminaccount' is a list while the 'i' is a variable name called when the loop has found the specific variable.

### 6.0 String operator

#### String isdigit() method

```
if age.isdigit():
```

*Figure 6.1*



As figure 6.1 above, 'isdigit' is to make sure that the value of the variable 'age' contains only digit.

### String upper case

```
name.upper()
```

*Figure 6.2*

In figure 6.2, using 'upper' behind the variable will make the alphabet values of the variable into uppercases.

### Strip and split

```
reclist = rec.strip().split('|')
```

*Figure 6.3*

This python '.strip' method will remove excessive spaces or a specific character from start to the end of the string. While '.split('|')' is to split the string into a list using '|' as a delimiter.

## 7.0 Control structure

### If

```
if len(adminpassword)<= 7:  
    print('Password is too short!!')  
    addadminaccount()
```

Figure 7.1

'if' is for the program to make a decision according to the condition. As figure 7.1 shown above, the given condition is the length of input variable 'adminpassword' is less than 7. If the condition match with the user's input. It will then show 'Password is too short!!' and call adminaccount function.

### If – else

```
if (success):  
    print('Login succesful!')  
    customermenu(loginlist)  
else:  
    print('Invalid account number and password!')  
    print('Please login again!')  
    logincustomer()
```

Figure 7.2

In figure 7.3, 'success' is a Boolean. The condition is if success has True value. If it has, not only it will break the loop, it will also display 'Login successful'. But if the condition of success is not achieved, it will run the code under the 'else'.

#### If – else if

```
if accounttype == '1':  
    logincustomer()  
elif accounttype == '2':  
    loginadmin()
```

Figure 7.3

'Elif' is used in python that also known as else if. In figure 7.3, the condition is if the input 'accounttype' equals to 1. If the condition is not match, it will then bring user to the second condition which is if the input equals to 2. With using else if, it can be different type of unlimited condition to run the specific code.

Nested – if

```
if phone_num.isdigit() == True:
    if len(phone_num) < 5:
        print("not a valid Phone Number")
```

Figure 7.4

In figure 7.4, nested if is used when the first condition is match. The first condition is if variable 'phone\_num' only contains digit. If True, it will then run another condition to check if the length of the variable is less than 5.

Nested – if – else

```
if age.isdigit():
    if int(age) <= 17:
        print('You are not eligible to create an account!')
        adminmenu()
    else:
        print('age registered!')
        break
```

Figure 7.5

Nested - if – else is a if else decision maker that falls under a big if condition. In figure 7.5, There is a condition to check if the user input variable 'age' only contains digits. If yes, then it will pass on to the smaller if where the condition is if the 'age' is less or equal to 17. If it is 17 or below, it will display 'You are not eligible to create an account!' Otherwise, it will then display 'age registered'.

Nested – if – else if

```
if phone_num.isdigit():
    if len(phone_num) < 8:
        print("Invalid Phone Number")
    elif len(phone_num) > 12:
        print("phone Number too long. Please try Again.")
```

Nested - if – else if is also a specific decision maker that falls under a big if condition. In figure 7.5, There is a condition to check if the user input variable 'phone\_num' only contains digits. If yes, then it will pass on to the smaller if where the condition is if the length of 'phone\_num' is less than 8. If it is less than 8, it will display 'Invalid phone number'. Otherwise, if it is more than 12, 'phone number too long. Please try again' will be shown.

## 8.0 Repetition structure

### For loop

```
for i in adminaccount:
```

*Figure 8.1*

'for' loop is used to repeat and read a variable such as list until a specific condition is achieved. In figure 8.1, the variable 'i' will be given value when the specific element is looped in the list 'adminaccount'.

### While loop

```
def menu():  
    while True:  
        print('-----Welcome to APU bank-----')  
        print('=====')
```

*Figure 8.2*

While loop is used to execute a block of statement repeatedly until a given condition is satisfied (Geekforgeeks, 2021). In figure 8.2, while loop is used for the menu function. While loop is used and not break so that the system is always ready for next customer.

Break

```

elif editchoice == '3':
    with open('Currentinfo.txt','w') as blank:
        blank.write('')
    with open('Currentinfo.txt','a') as saving:
        for edited_list in allrec:
            joinstring = '|'.join(edited_list)
            saving.write(joinstring + '\n')
    adminmenu()
    break

```

*Figure 8.3*

The 'break' is used to stop the repeating loop after the condition given is achieved. In figure 8.3, this is save and exit option for editing customer's detail function. When the admin inputs 3 it will then save the edited info and overwrite the text file. After those are performed, it will then stop the loop.

Continue

```

while True:
    editchoice = input('Choose a detail to edit \n1.Age \n2.Phone number \n3.Save and exit \n: ')
    if editchoice == '1':
        newage = input('Please enter new age: ')
        reclist[1] = newage
        print('Age has been changed successfully!')
        print('New customer detail:' + str(reclist))
        allrec[num] = reclist
        continue

```

*Figure 8.4*

If 'Continue' is used, it will stop the loop and return the loop to the beginning of the loop. In figure 8.4 'continue' is used in editing customer's detail function. After the admin edit 1 of the customer's details, it will restart the loop so that admin can either continue to edit or save and exit when he's done.

## 9.0 Exception handling

### Try and except

```
try:
    withdrawamount = int(input('Enter withdraw amount:'))
except ValueError:
    print ('Enter numbers only!')
```

*Figure 9.1*

Try and except work as a testing block and prevent errors. In figure 9.1, the customer will be inputting integers only for the variable 'withdrawamount' unless there is an error which is 'ValueError'.

Therefore, customer will not be able to enter value other than integer. Or else, it will display 'Enter numbers only!' to remind the customer.

## 10.0 Files

```
with open('Customerlogin.txt','r') as customerlogin:
```

*Figure 10.1*

In figure 10.1 shown above, 'open' is used to create a file and make it as a variable named 'customerlogin'. This text file is open in read mode; therefore it cannot perform other function than just reading and acquiring data.

### Append

```
with open('Savinginfo.txt','a') as saving:
```

*Figure 10.2*

In figure 10.3, this text file is opened in append mode. It is also made to a variable named 'saving'. In append mode, the program will add or append data into this text file instead of overwriting while using write function

Write

```
customerlogin.write(rec)
```

*Figure 10.3*

‘Write’ is used to either append or overwriting depending on the text file opened. In figure 10.3, the variable ‘rec’ is being append into the ‘customerlogin’ variable that has a value of a text file in append mode.

11.0 Module

```
import datetime
```

*Figure 11.1*

```
date = datetime.datetime.now()
```

*Figure 11.2*

A module is a readymade code that is available for programmers to import. In Figure 11.1, ‘import’ is used to import ‘datetime’ module. It is then able to use like shown in figure 11.2. This datetime module is to assign a present date to the variable ‘date’.

12.0 Function

Function name	Python code	Purposes
menu	<pre>def menu()</pre>	For users to login as admins, super user or customer
customermenu	<pre>def customermenu(loginlist)</pre>	Menu for customers to show what different choices they can perform
Adminmenu	<pre>def adminmenu()</pre>	Menu for admins to show what action they can perform

Supermenu	<code>def supermenu()</code>	Menu for super users to show what action they can perform
Viewcustomerdetail	<code>def viewcustomerdetail()</code>	A function for admin to check on every customer's detail
Balanceinquiry	<code>def balanceinquiry(loginlist)</code>	For customer to check their balance and how many is available for withdrawal
loginadmin	<code>def loginadmin()</code>	For admin and super users to login
Logincustomer	<code>def logincustomer()</code>	For customers to login either saving customer or current customer
addAdminAccount	<code>def addadminaccount()</code>	For super user to create admin account
Changecustomerpassword	<code>def changecustomerpassword(loginlist)</code>	For customer to change their password to login
addSavingAccount	<code>def addSavingAccount()</code>	For admins to create saving account
addCurrentAccount	<code>def addCurrentAccount()</code>	For admins to create current account
savingwithdrawal	<code>def savingwithdrawal(loginlist)</code>	For saving customers to withdraw money
Currentwithdrawal	<code>def currentwithdrawal(loginlist)</code>	For current customers to withdraw money
Savingdeposit	<code>def savingdeposit(loginlist)</code>	For saving customers to deposit money
currentdeposit	<code>def currentdeposit(loginlist)</code>	For current customers to deposit money

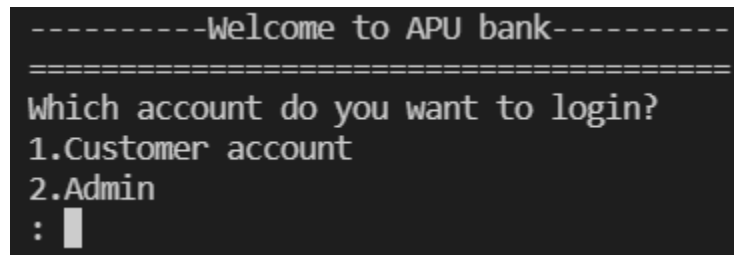


Editsavingdetail	<code>def editsavingdetail()</code>	For admins to edit saving customer details
Editcurrentdetail	<code>def editcurrentdetail()</code>	For admins to edit current customer details
transaction	<code>def transaction()</code>	For super users and admins to get customers transaction from time to time

## Screenshots of samples and explanation

### 1.0 Menu

```
-----Welcome to APU bank-----  
=====
```



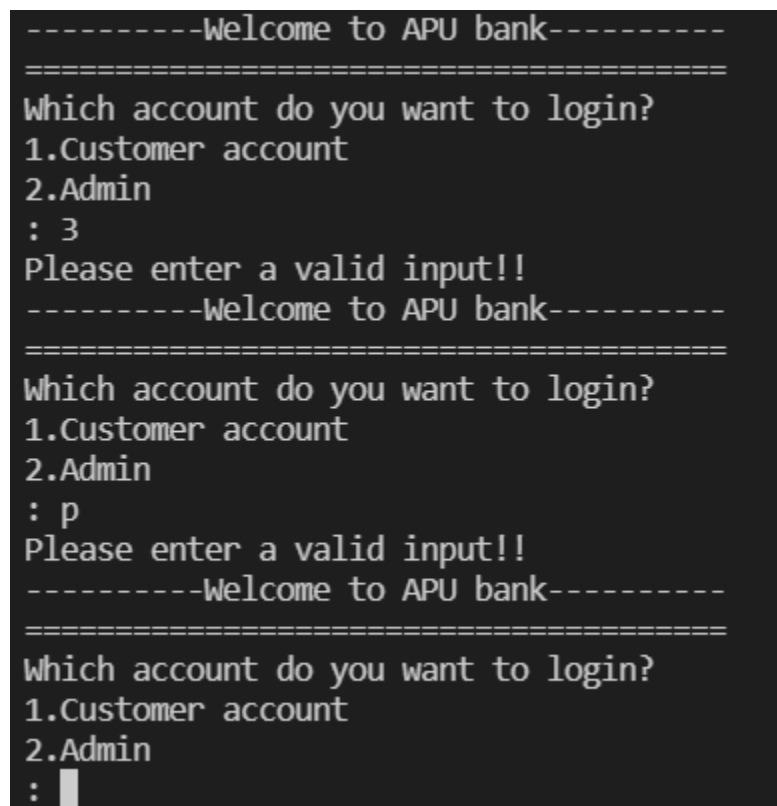
```
Which account do you want to login?  
1.Customer account  
2.Admin  
: █
```

*Figure 1.1*

This is the first menu for every system user including saving customers, current customers, admins, and super user. In this menu, customers can login by inputting 1 while inputting 2 is for admins and super user.

### 2.0 Menu wrong inputs

```
-----Welcome to APU bank-----  
=====
```



```
Which account do you want to login?  
1.Customer account  
2.Admin  
: 3  
Please enter a valid input!!  
-----Welcome to APU bank-----  
=====
```

```
Which account do you want to login?  
1.Customer account  
2.Admin  
: p  
Please enter a valid input!!  
-----Welcome to APU bank-----  
=====
```

```
Which account do you want to login?  
1.Customer account  
2.Admin  
: █
```

*Figure 2.1*

In this figure above, user can only input '1' and '2'. Other than those, it would display 'Please enter a valid input'. Then it will loop from the start again for every user to re-enter their inputs.

### 3.0 Login successful

```
-----Welcome to APU bank-----  
=====
```

Which account do you want to login?

- 1.Customer account
- 2.Admin

: 1

Please enter your account number

: 100001

Please enter your password

: ppap

Login succesful!

```
-----You are in customer menu-----  
=====
```

Please choose an option

- 1.Withdrawal
- 2.Deposit
- 3.Balance Inquiry
- 4.Change password
- 5.Exit

:

*Figure 3.1*

```
-----Welcome to APU bank-----  
=====
```

Which account do you want to login?

- 1.Customer account
- 2.Admin

: 2

Please enter your username

: lee

Please enter your password

: 123123123

Admin login succesful!

```
-----You are in admin menu-----  
=====
```

Please choose an option

- 1.Create customer account
- 2.Edit customer detail
- 3.View all data
- 4.Transaction
- 5.Exit

: █

*Figure 3.2*

```
-----Welcome to APU bank-----  
=====
```

Which account do you want to login?

- 1.Customer account
- 2.Admin

: 2

Please enter your username

: admin

Please enter your password

: admin

```
-----You are in super user account menu-----  
=====
```

Please choose an option

- 1.Create admin account
- 2.Transaction
- 3.Exit

: █

*Figure 3.3*

In the figures 3.0, 3.1, 3.2 above are the examples of login successful for each user. If the login attempts are successful, they will then be brought to the menus accordingly.

#### 4.0 Fail login attempts

```
-----welcome to APU bank-----  
=====
```

Which account do you want to login?  
1.Customer account  
2.Admin  
: 1  
Please enter your account number  
: 100003  
Please enter your password  
: 000  
Invalid account number and password!  
Please login again!

*Figure 4.1*

```
-----welcome to APU bank-----  
=====
```

Which account do you want to login?  
1.Customer account  
2.Admin  
: 2  
Please enter your username  
: lee  
Please enter your password  
: 222  
Invalid admin username and password!  
Please login again!

*Figure 4.2*

These are the examples of login fail attempts. Users will be asked to login again.

## 5.0 Customer menu

```
-----You are in customer menu-----  
=====
```

Please choose an option

- 1.Withdrawal
- 2.Deposit
- 3.Balance Inquiry
- 4.Change password
- 5.Exit

:

*Figure 5.1*

In this figure above, is the menu for customers including saving and current customers. There are 5 choices available for them which are withdrawal, deposit, balance inquiry, change password, exit. They will need to enter the correct input accordingly to access the functions.

## 6.0 Withdrawal

```
-----You are in customer menu-----  
=====
```

Please choose an option

- 1.Withdrawal
- 2.Deposit
- 3.Balance Inquiry
- 4.Change password
- 5.Exit

: 1

Withdraw from

- 1.Saving
- 2.Current
- 3.Exit

: 2

```
-----Withdraw from current-----  
=====
```

Your balance is =RM750

Enter withdraw amount:200

Your balance is : RM550

*Figure 6.1*

```

-----You are in customer menu-----
=====
Please choose an option
1.Withdrawal
2.Deposit
3.Balance Inquiry
4.Change password
5.Exit
: 1
Withdraw from
1.Saving
2.Current
3.Exit
: 1
-----Withdraw from Savings-----
=====
Your balance is:RM 350
Enter withdraw amount:10
Your balance is : RM340

```

Figure 6.2

```

-----Withdraw from Savings-----
=====
Your balance is:RM 1600
Enter withdraw amount:1600
Withdraw unsuccessful!
Balance insufficient!
Withdraw from
1.Saving
2.Current
3.Exit
: █

```

Figure 6.1

This is the page for customers to withdraw. Customers will need to pick the account withdrawing from accordingly. Then it will show their balance and the amount of withdrawal available for them. Customers are not able to withdraw more than the available withdrawal. If attempt is done, withdraw will not be performed. Saving and current customers have different minimum balance for them to perform any withdrawal. Saving's account minimum balance is RM100 while current ones is RM500.

## 7.0 Withdrawal wrong input

```
Withdraw from
1.Saving
2.Current
3.Exit
: 2
-----Withdraw from current-----
=====
Withdraw from
1.Saving
2.Current
3.Exit
: █
```

*Figure 7.1*

```
-----Withdraw from Savings-----
=====
Your balance is:RM 1590
Enter withdraw amount:p
Enter numbers only!
Enter withdraw amount:█
```

*Figure 7.2*

If a saving or current customer picks the opposite account to withdraw, they will not be able to access. They can only withdraw from their own account accordingly. Besides, If the withdraw amount is not integer, customers will be told to enter the withdraw amount again.



## 8.0 Deposit

```
-----You are in customer menu-----  
=====
```

Please choose an option

- 1.Withdrawal
- 2.Deposit
- 3.Balance Inquiry
- 4.Change password
- 5.Exit

: 2

Deposit to

- 1.Saving
- 2.Current
- 3.Exit

: 1

```
-----Deposit to Savings-----  
=====
```

Your balance is:RM 340  
Enter deposit amount:1250  
Your balance is : RM1590

*Figure 8.1*

```
-----Deposit to current-----  
=====
```

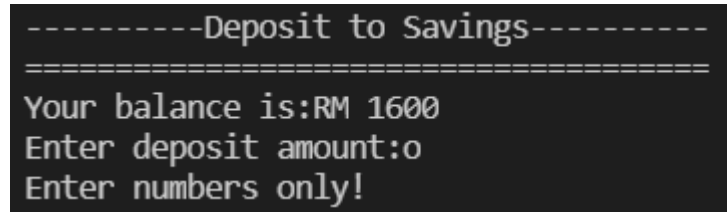
Your balance is:RM 850  
Enter deposit amount:10  
Your balance is : RM860

*Figure 8.2*

In deposit function, examples are shown in figures above. Customer will pick the account type accordingly and input the amount of deposit. It will then show the balance after depositing.

### 9.0 Deposit wrong input

```
-----Deposit to Savings-----  
=====
```



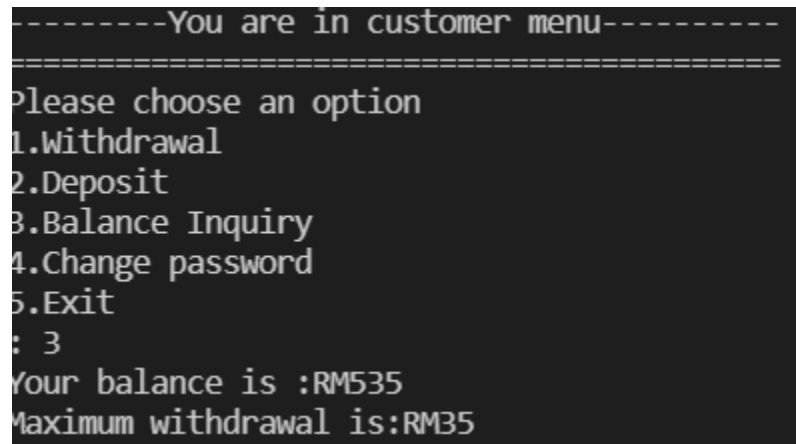
```
Your balance is:RM 1600  
Enter deposit amount:o  
Enter numbers only!
```

*Figure 9.1*

This figure shows that only numbers can be input while depositing an amount into their account. Input other than integers will display ‘Enter numbers only!’ to remind the customers.

### 10.0 Balance inquiry

```
-----You are in customer menu-----  
=====
```



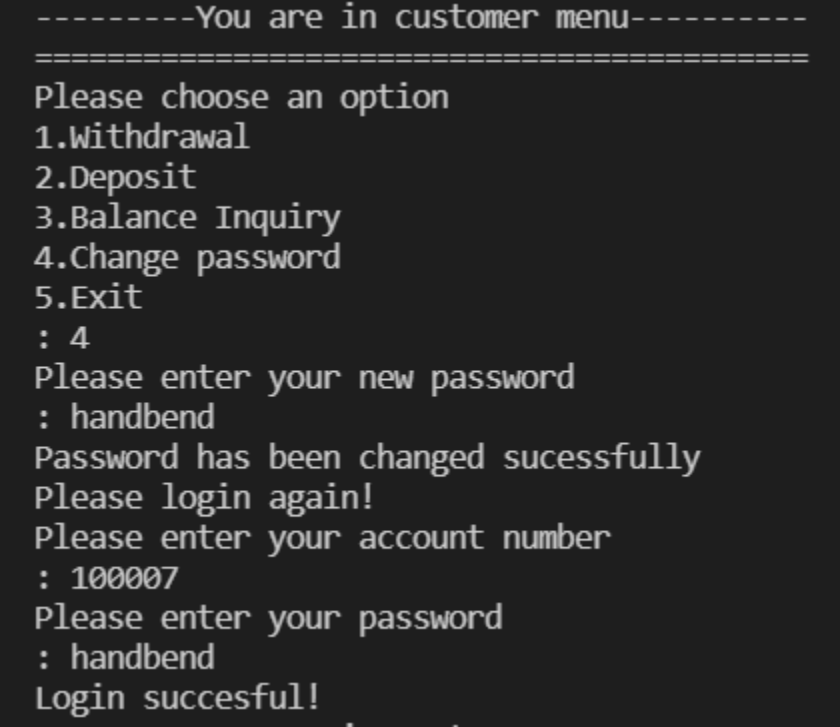
```
Please choose an option  
1.Withdrawal  
2.Deposit  
3.Balance Inquiry  
4.Change password  
5.Exit  
: 3  
Your balance is :RM535  
Maximum withdrawal is:RM35
```

*Figure 10.1*

This function is to show the balance of their account and the maximum amount is available to withdraw.

## 11.0 Change password

```
-----You are in customer menu-----  
=====
```



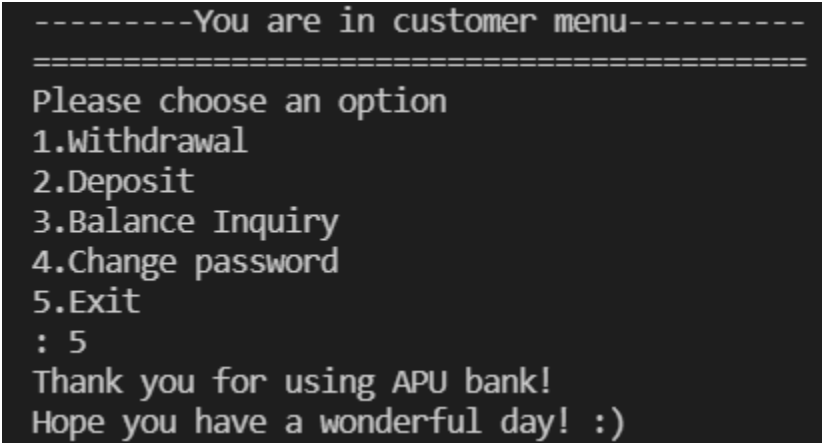
```
Please choose an option  
1.Withdrawal  
2.Deposit  
3.Balance Inquiry  
4.Change password  
5.Exit  
: 4  
Please enter your new password  
: handbend  
Password has been changed sucessfully  
Please login again!  
Please enter your account number  
: 100007  
Please enter your password  
: handbend  
Login succesful!
```

*Figure 11.1*

This function is for customers to change their password that is given during the account was created. Every customer will be given 'default' as default password to login before the password is changed. After changing the password, they will be asked to login again.

## 12.0 Exit

```
-----You are in customer menu-----  
=====
```



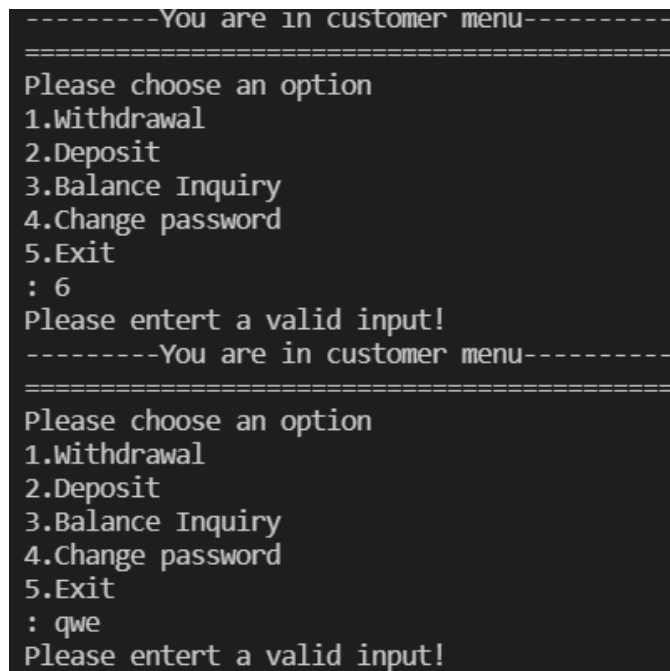
```
Please choose an option  
1.Withdrawal  
2.Deposit  
3.Balance Inquiry  
4.Change password  
5.Exit  
: 5  
Thank you for using APU bank!  
Hope you have a wonderful day! :)
```

*Figure 12.1*

Exit is the 5<sup>th</sup> option of this customer menu. When the customer is done, they can press this to exit or log out to the main menu for the next user.

### 13.0 Customer menu wrong input

```
-----You are in customer menu-----  
=====
```



```
Please choose an option  
1.Withdrawal  
2.Deposit  
3.Balance Inquiry  
4.Change password  
5.Exit  
: 6  
Please enter a valid input!  
-----You are in customer menu-----  
=====
```

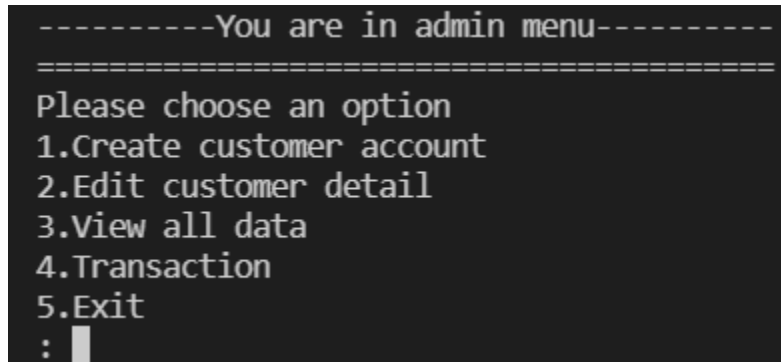
```
Please choose an option  
1.Withdrawal  
2.Deposit  
3.Balance Inquiry  
4.Change password  
5.Exit  
: qwe  
Please enter a valid input!
```

*Figure 13.1*

As shown in figure above, inputs other than '1', '2', '3', '4', '5' it will print 'Please enter a valid input' and loop.

### 14.0 Admin menu

```
-----You are in admin menu-----  
=====
```



```
Please choose an option  
1.Create customer account  
2.Edit customer detail  
3.View all data  
4.Transaction  
5.Exit  
: █
```

*Figure 14.1*

This figure shows the menu for admin. When admin has logged in successfully, they will be brought here. There are 5 choices for them to choose by inputting '1', '2', '3', '4', '5' which are creating customer account, edit customer detail, view all data, view transaction and exit.

## 15.0 Create customer account

```
-----You are in admin menu-----  
=====
```

Please choose an option

- 1.Create customer account
- 2.Edit customer detail
- 3.View all data
- 4.Transaction
- 5.Exit

: 1

Which account do you want to create?

- 1.Saving
- 2.Current
- 3.Exit

: 1

Please enter your name

: Bob

Name registered!

Please enter your age

: 44

age registered!

Please input your phone number

: 0123558346

Phone number registered!.

Please enter your ic number

: 9909190935

IC less than 12 digit, Please enter again!

Please enter your ic number

: 990919113535

IC number registered!

Account has been made!

Your account info:Bob|44|0123558346|990919113535

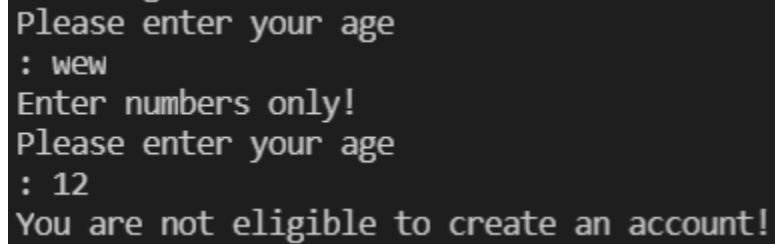
This is your account number:100008

Your password is"default", Please change immediately!

*Figure 15.1*

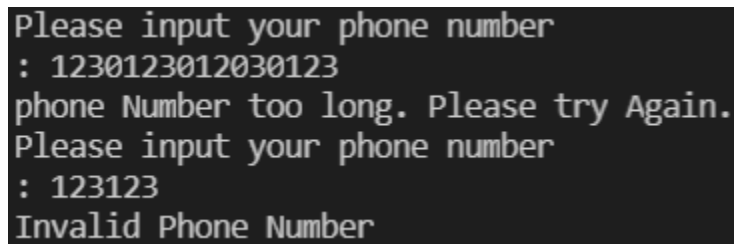
Create customer account is the 1<sup>st</sup> accessible function for admin staff only. After inputting 1 in the admin menu, it will ask the admin 'which account do you want to create?' Then admin will pick either saving or current account. Then, admin will have to register the account according to the customer's info which is name, age, phone number and IC number. The figure above is an example of entering customer's info and a mistake was made by the admin while entering IC number.

## 16.0 Create customer account wrong input



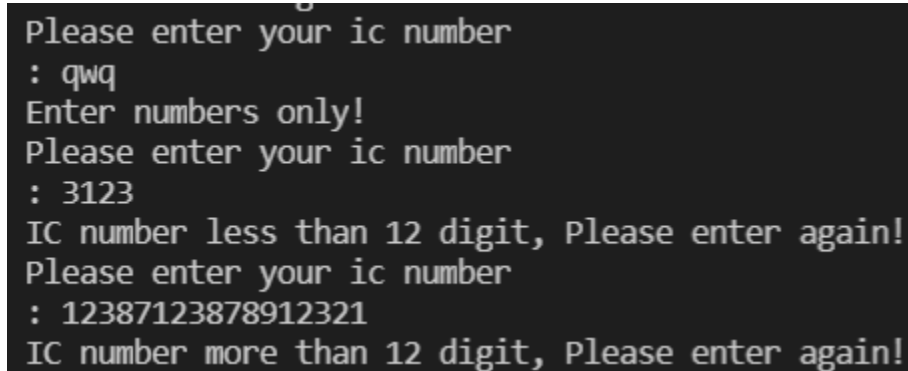
```
Please enter your age
: weW
Enter numbers only!
Please enter your age
: 12
You are not eligible to create an account!
```

*Figure 16.1*



```
Please input your phone number
: 1230123012030123
phone Number too long. Please try Again.
Please input your phone number
: 123123
Invalid Phone Number
```

*Figure 16.2*



```
Please enter your ic number
: qwq
Enter numbers only!
Please enter your ic number
: 3123
IC number less than 12 digit, Please enter again!
Please enter your ic number
: 12387123878912321
IC number more than 12 digit, Please enter again!
```

*Figure 16.3*

As figures shown above, these are the words displayed when the system accepts wrong inputs while creating customer account. In these 3 customers info, input value other than integer are not accepted. Besides, Customer under 17 are not eligible to create an account. The length of phone number cannot be less than 8 numbers and more than 10 numbers. For IC number, input length less than 12 and more than 12 will also not be accepted.

## 17.0 Edit customer detail

```
-----You are in admin menu-----  
=====
```

Please choose an option

- 1.Create customer account
- 2.Edit customer detail
- 3.View all data
- 4.Transaction
- 5.Exit

: 2

Which account do you want to edit?

- 1.Saving
- 2.Current
- 3.Exit

: 1

Please enter account number to edit

: 100001

Account found!

Account: ['logan', '19', '0163353828', '0210031101249', '100001', '1600']

Choose a detail to edit

- 1.Age
- 2.Phone number
- 3.Save and exit

: 1

Please enter new age: 22

Age has been changed successfully!

New customer detail:['logan', '22', '0163353828', '0210031101249', '100001', '1600']

Choose a detail to edit

- 1.Age
- 2.Phone number
- 3.Save and exit

: 3

*Figure 17.1*

In this figure above is an example of how to edit customer detail. First, admin will choose an account type and search across the text file by inputting the account number accordingly. When account is found, the account detail will be shown. There are only 2 details can be edited by admin which are age and phone number. 'Age has been changed successfully!' and the edited customer detail will be displayed after the edit is done. Then, admin can check if any mistake or make another change before saving and exit from this function.

### 18.0 Edit customer detail wrong input

```
Account: ['logan', '19', '0163353828', '0210031101249', '100001', '1600']
Choose a detail to edit
1.Age
2.Phone number
3.Save and exit
: 1
Please enter new age: po
Enter numbers only!
```

*Figure 18.1*

```
Account: ['logan', '19', '0163353828', '0210031101249', '100001', '1600']
Choose a detail to edit
1.Age
2.Phone number
3.Save and exit
: 2
Please enter new phone number: we
Enter numbers only!
```

*Figure 18.2*

As shown in figures above, while admin is editing customer detail, it is very similar to the registration of customer account. Age and phone number cannot be other values except integer.



19.0 View all data

```

-----You are in admin menu-----
=====
Please choose an option
1.Create customer account
2.Edit customer detail
3.View all data
4.Transaction
5.Exit
: 3
Which account detail do you want to view?
1.Saving
2.Current
3.Exit
: 1
  Name |Age|Phone no| IC number |Acc no|Balance
=====
logan|19|0163353828|021003101249|100001|1590

zekang|20|0128889870|023492930123|100002|900

jou|20|0122391860|010410101235|100005|623

colin|19|01123893450|020615109977|100006|0

bob|44|0123558346|990919113535|100008|0

```

*Figure 19.1*

```

Which account detail do you want to view?
1.Saving
2.Current
3.Exit
: 2
  Name |Age|Phone no| IC number |Acc no|Balance
=====
darren|19|0182126338|9999999999|100003|850

johnny|20|0162765035|021003101249|100004|750

kah xiang|19|01234567890|021003109977|100007|535

```

*Figure 19.2*

This function is to view all account details. Admin will get to choose what type of account detail they want to check. Then the file will read every existing account.

## 20.0 View transactions

```

-----You are in admin menu-----
=====
Please choose an option
1.Create customer account
2.Edit customer detail
3.View all data
4.Transaction
5.Exit
: 4
Please enter your account number: 100007
Please enter start day: 10
Please enter start month: 12
Please enter start year: 2021
Please enter end day: 10
Please enter end month: 12
Please enter end year: 2021
10/12/2021
10/12/2021
['kah xiang', '100007', 'withdraw', 'RM1', '10/12/2021']
['kah xiang', '100007', 'deposit', 'RM750', '10/12/2021']
['kah xiang', '100007', 'withdraw', 'RM200', '10/12/2021']
['kah xiang', '100007', 'withdraw', 'RM15', '10/12/2021']
Total withdrawal: RM216
Total deposit: RM750

```

*Figure 20.1*

This transaction function is to generate customer's statement of account report. To use it, admin will have to first enter the account number that he is searching for. Then he will need to enter according the duration he is seeking for from start day, start month, start year, end day, end month, end year. Then the statement will be displayed with some customer info so that is convenience for admins to identify. Also, the total of withdrawal and deposit will also be accumulated in the end.

## 21.0 View transaction wrong inputs

```

-----You are in admin menu-----
=====
Please choose an option
1.Create customer account
2.Edit customer detail
3.View all data
4.Transaction
5.Exit
: 4
Please enter your account number: wqe
Enter numbers only!

```

*Figure 21.1*

```

Please enter start day(dd): qwe
Please enter start month(mm): wqe
Please enter start year(yyyy): we
Please follow the format (dd for day) (MM for month) (yyyy for year)

```

*Figure 21.2*

In this figure, wrong inputs are shown. While entering the account number, it only accepts numbers. If not, the admin will have to enter again. So as the start day, start month and start year. The system also show how is input supposed to be.

## 22.0 Exit

```

-----You are in admin menu-----
=====
Please choose an option
1.Create customer account
2.Edit customer detail
3.View all data
4.Transaction
5.Exit
: 5
Thank you for using APU bank!
Hope you have a wonderful day! :)

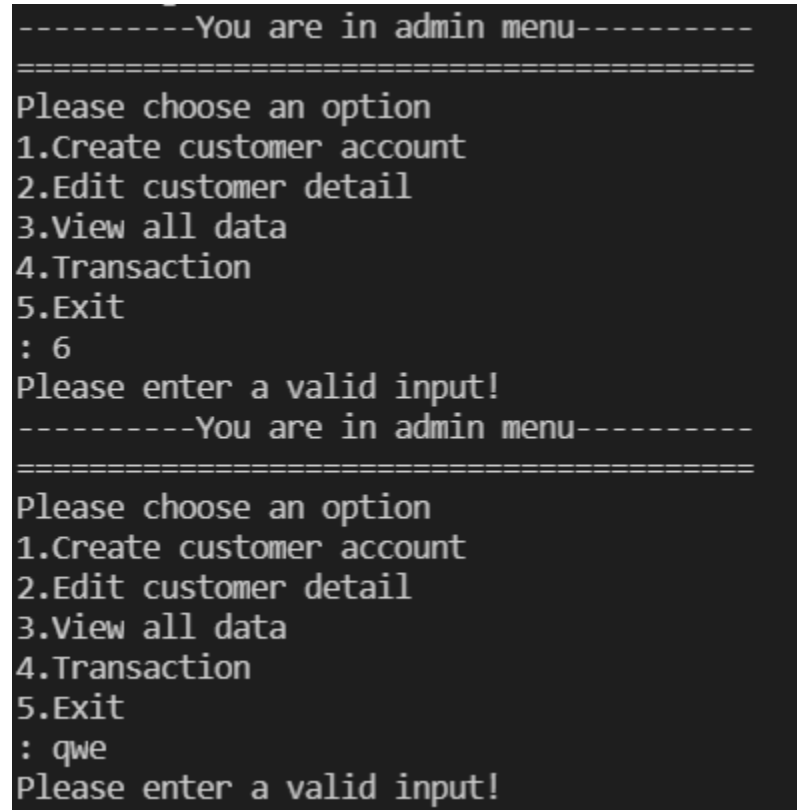
```

*Figure 22.1*

After the admin is done, he or she can enter an input of '5'. Then it will return back to the main menu.

### 23.0 Admin menu wrong inputs

```
-----You are in admin menu-----  
=====
```



```
Please choose an option  
1.Create customer account  
2.Edit customer detail  
3.View all data  
4.Transaction  
5.Exit  
: 6  
Please enter a valid input!  
-----You are in admin menu-----  
=====
```

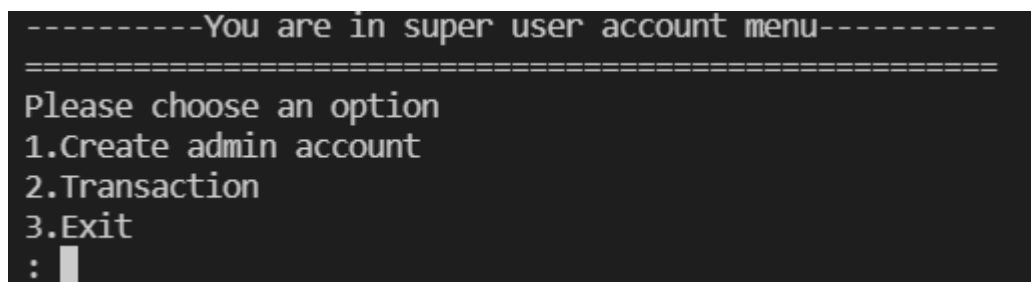
```
Please choose an option  
1.Create customer account  
2.Edit customer detail  
3.View all data  
4.Transaction  
5.Exit  
: qwe  
Please enter a valid input!
```

*Figure 23.1*

In admin menu, input other than '1','2','3','4','5' are not accepted as a valid input. It will loop forever until a valid input is detected from the user.

### 24.0 Super menu

```
-----You are in super user account menu-----  
=====
```



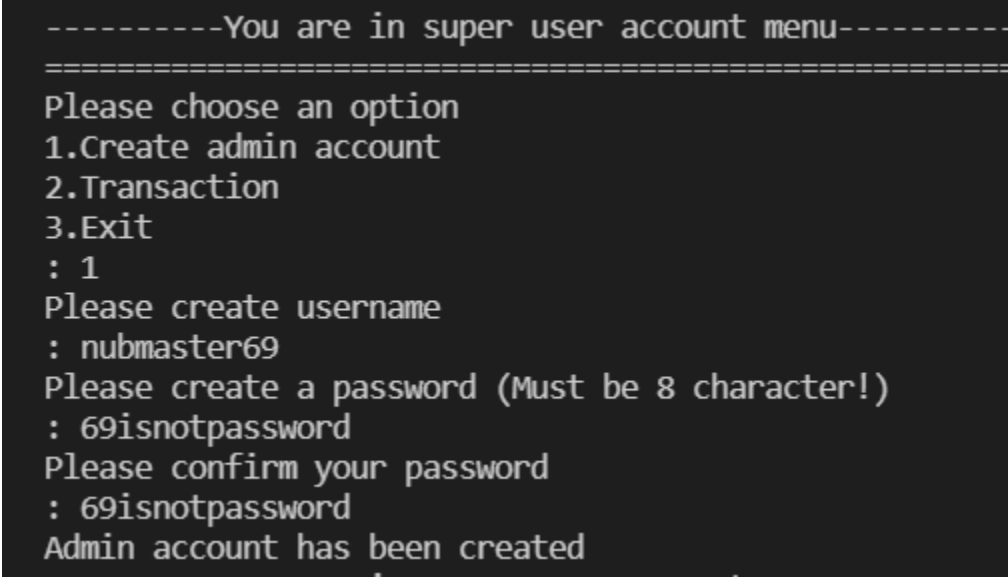
```
Please choose an option  
1.Create admin account  
2.Transaction  
3.Exit  
: █
```

*Figure 24.1*

The figure above shows the menu for super user. Only super user that know the super user username and password will have access to this menu.

## 25.0 Create admin account

```
-----You are in super user account menu-----  
=====
```

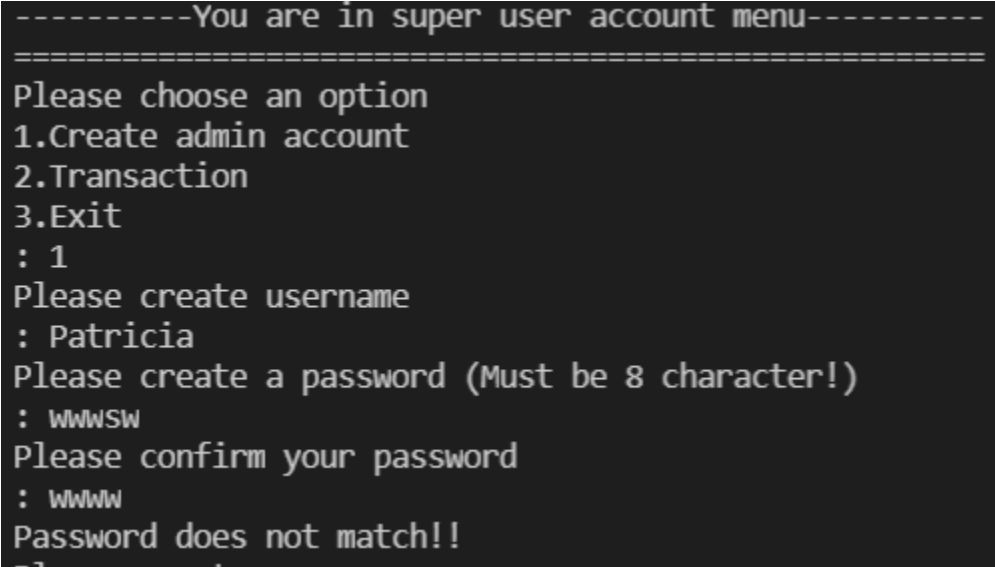


A terminal window with a black background and white text. It shows a menu for a super user account. The user selects option 1, 'Create admin account'. They are prompted to create a username and enter 'nubmaster69'. They are then prompted to create a password (must be 8 characters) and enter '69isnotpassword'. They confirm the password by entering '69isnotpassword' again. The final message is 'Admin account has been created'.

```
Please choose an option  
1.Create admin account  
2.Transaction  
3.Exit  
: 1  
Please create username  
: nubmaster69  
Please create a password (Must be 8 character!)  
: 69isnotpassword  
Please confirm your password  
: 69isnotpassword  
Admin account has been created
```

*Figure 25.1*

```
-----You are in super user account menu-----  
=====
```



A terminal window with a black background and white text. It shows the same menu as Figure 25.1. The user selects option 1, 'Create admin account'. They are prompted to create a username and enter 'Patricia'. They are then prompted to create a password (must be 8 characters) and enter 'WWW SW'. They confirm the password by entering 'WWW W'. The final message is 'Password does not match!!'.

```
Please choose an option  
1.Create admin account  
2.Transaction  
3.Exit  
: 1  
Please create username  
: Patricia  
Please create a password (Must be 8 character!)  
: WWW SW  
Please confirm your password  
: WWW W  
Password does not match!!
```

*Figure 25.2*

There are only 2 functions for super user which is create an admin account and view customers' transaction statement.

## 26.0 View transactions

```
-----You are in super user account menu-----  
=====
```

Please choose an option

- 1.Create admin account
- 2.Transaction
- 3.Exit

: 2

Please enter your account number: 100001

Please enter start day: 10

Please enter start month: 12

Please enter start year: 2021

Please enter end day: 10

Please enter end month: 12

Please enter end year: 2021

10/12/2021

10/12/2021

```
['logan', '100001', 'withdraw', 'RM10', '10/12/2021']  
['logan', '100001', 'deposit', 'RM1250', '10/12/2021']  
['logan', '100001', 'withdraw', 'RM10', '10/12/2021']  
['logan', '100001', 'deposit', 'RM20', '10/12/2021']  
['logan', '100001', 'deposit', 'RM0', '10/12/2021']  
Total withdrawal: RM20  
Total deposit: RM1270
```

*Figure 26.1*

This transaction function is same as the one that is accessible for admin.

## 27.0 View transaction wrong input

```
-----You are in admin menu-----  
=====
```

Please choose an option  
1.Create customer account  
2.Edit customer detail  
3.View all data  
4.Transaction  
5.Exit  
: 4  
Please enter your account number: wqe  
Enter numbers only!

Figure 27.1

```
Please enter start day(dd): qwe  
Please enter start month(mm): wqe  
Please enter start year(yyyy): we  
Please follow the format (dd for day) (MM for month) (yyyy for year)
```

Figure 27.2

While trying to enter value other than integer, it will result in figures above. It will give instructions for admin to enter the correct input accordingly.

## 28.0 Exit

```
-----You are in super user account menu-----  
=====
```

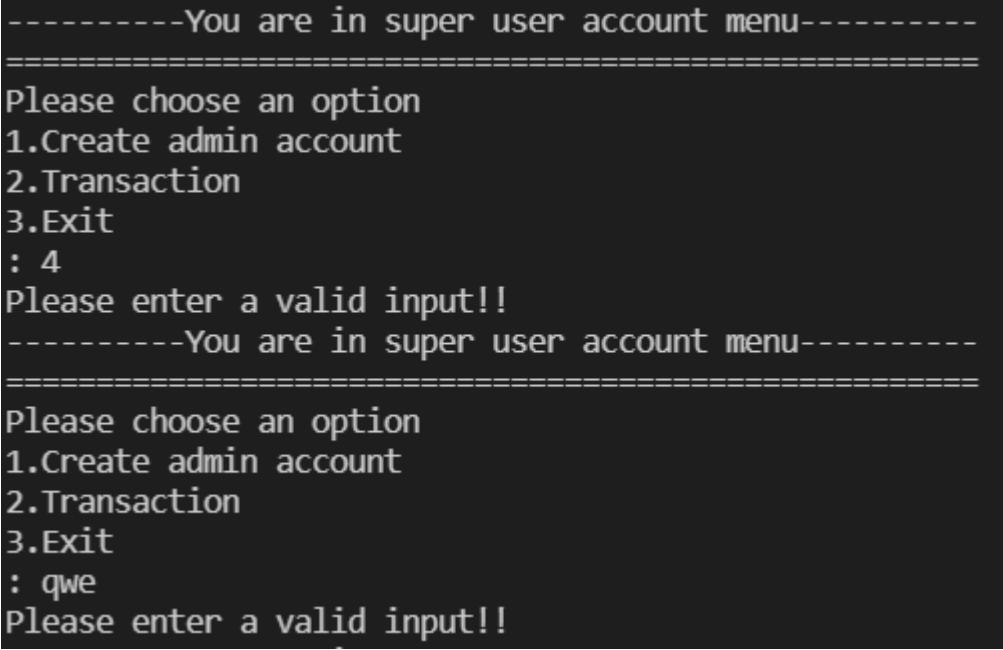
Please choose an option  
1.Create admin account  
2.Transaction  
3.Exit  
: 3  
Thank you for using APU bank!  
Hope you have a wonderful day! :)

Figure 28.1

When super user is done with the job, he or she can enter '3' as an input to log out from super user. Then, the system will return to the main menu and ready for the next user.

## 29.0 Super menu wrong inputs

```
-----You are in super user account menu-----  
=====
```



```
Please choose an option  
1.Create admin account  
2.Transaction  
3.Exit  
: 4  
Please enter a valid input!!  
-----You are in super user account menu-----  
=====
```

```
Please choose an option  
1.Create admin account  
2.Transaction  
3.Exit  
: qwe  
Please enter a valid input!!
```

*Figure 29.1*

In this figure, when input other than '1', '2', '3' is received in this menu, user is asked to enter a valid input



## **Conclusion**

As a conclusion, this assignment not only help me gain knowledge regarding python programming, but also improve my logical thinking towards programming language. Practice makes perfect, I have faced many challenging errors and overcome while doing python programming. It is known that python is one of the easiest programming languages. However, I still managed to work hard enough to handle basic functions for specific purposes and complete the bank system that is usable. There are always room for improvement for me to master python programming. Therefore, I am looking forward about learning more about programming even if it is more challenging.

## **Reference**

Chugh, A. (2020, May 28). *10 Efficient Ways to Use Python Lists - Better Programming*. Medium.

<https://betterprogramming.pub/10-efficient-ways-to-use-python-lists-f6e7e666708>

GeeksforGeeks. (2020, August 25). *Difference between Source Code and Object Code*.

<https://www.geeksforgeeks.org/difference-between-source-code-and-object-code/#:~:text=Source%20code%20refers%20to%20high,is%20generated%20by%20human%2Fprogrammer.&text=In%20simple%20we%20can%20say,%2C%20Python%2C%20Assembly%20language%20etc.>

GeeksforGeeks. (2021, December 7). *Reading and Writing to text files in Python*.

<https://www.geeksforgeeks.org/reading-writing-text-files-python/>