TABLE OF CONTENTS

1.0 Introduction

The designed program is a COVID-19 Patient Management System. The main purpose is to record patients' information and their status automatically instead of drawing a table and typing the information manually using Microsoft Word. The program has the function of registering new patients' information, recording test results and action taken, modifying patients' status, showing statistical information and searching patients' data.

The program is expected to accept the input of users according each condition and record the information in each text file. It will be only available for patients who belong to one group(ATO, ACC, AEO, SID, AHS) and one zone(A, B, C, D). Group, zone and action taken for COVID-19 positive patients will be recorded using the following abbreviation.

Group/Zone/Action Taken	**Abbreviation**
Asymptomatic individuals with history of travelling overseas	ATO
Asymptomatic individuals who has close contact with positive patients	ACC
Asymptomatic individuals who had attended event associated with known COVID-19 outbreak	AEO
Symptomatic individuals	SID
Asymptomatic hospital staff	AHS
East	A
West	B
North	C
South	D
Continue Working (Follow-Up Test Required)	CWFR
Home Quarantine (Follow-Up Test Required)	HQFR
Quarantine in Designated Centres (Follow-Up Test Required)	QDFR
Home Quarantine (No Follow-Up Test Required)	HQNF
Quarantine in Hospital Normal Ward or ICU (No Follow-Up Test Required)	QHNF
Allow to reunion with family	RU
Continue Working	CW

Table 1.1 Meaning of all the Abbreviations

In this program, the patients will be required to complete three tests in 14 days to prove that he/she is totally free from COVID-19. All the patients will go through test1 as their first test and the follow-up tests will be done in series. Patients with positive test result in test1 or test2 will not need to do the following test. Patient ID and case ID will be in sequence and will include their group and zone.

Information of registered patients include patient ID, name, age, group, zone, contact number and email address will be recorded in a text file. Patients' test details and action taken will be recorded in six text files ——— three text files for negative test result and three text files for positive test result according each test (1, 2, 3). Case ID, patient ID, zone, group and status of patients who test positive for COVID-19 will be recorded in a text file.

In the program, there will be around 20%-30% of patients test positive for COVID-19 in each test. Around 50% of patients with positive test result will be remained as active cases, around 30% of patients will be changed to recovered cases and around 20% of patients will be changed to deceased cases.

2.0 Design of Program

2.1 Pseudocode

```
PROGRAM COVID-19PatientManagementSystem
BEGIN
FUNCTION searchPatient(fileName,patientName)
    OPEN and READ fileName as fileHandler
    FOR EACH line IN fileHandler
        IF (patientName IN line) THEN
            CLOSE fileHandler
            RETURN FALSE
        ENDIF
    ENDFOR
    CLOSE fileHandler
    RETURN TRUE
ENDFUNCTION

FUNCTION patientRegistration()
    DOWHILE TRUE
        patient=[]
        patientName = READ("Patient Name,(x) to exit: ") in UPPER case as STRING
        IF (patientName='X') THEN
            BREAK
        ENDIF
        IF(searchPatient(Patient_Detail.txt,patientName)) THEN
            OPEN and READ Patient_Detail.txt as getID
            count=0
            FOR EACH line IN getID
                count=count+1
            ENDFOR
            count=count+1
            CLOSE Patient_Detail.txt

            age = READ("Age:")
            IF (age is an integer) THEN
                READ age as STRING
            ELSE
                PRINT("Invalid age.")
                BREAK
            ENDIF

            PRINT("Select a group:")
            PRINT("ATO=Asymptomatic individuals with history of travelling overseas")
            PRINT("ACC=Asymptomatic individuals with history of contact with known case of
COVID-19")
            PRINT("AEO=Asymptomatic individuals who had attended event associated with known
COVID-19 outbreak")
            PRINT("SID=Symptomatic individuals")
            PRINT("AHS=Asymptomatic hospital staff")
            group = READ("Group: ") in UPPER case as STRING
            IF (group='ATO' or group='ACC' or group='AEO' or group='SID' or group='AHS')
THEN
                PASS
            ELSE
                PRINT("Invalid group.")
                BREAK
            ENDIF

            PRINT("Select a zone:")
            PRINT("A-East")
            PRINT("B-West")
            PRINT("C-North")
            PRINT("D-South")
            zone=READ("Zone(A/B/C/D):")in UPPER case as STRING
            IF (zone='A' or zone='B' or zone='C' or zone='D) THEN
                PASS
            ELSE
                PRINT("Invalid zone.")
                BREAK
            ENDIF

            contactNumber = READ("Contact Number:")
            IF (contactNumber is an integer) THEN
                contactNumber = '0' + contactNumber
                READ contactNumber as STRING
            ELSE
            PRINT("Invalid contact number.")
                BREAK
            ENDIF

            emailAddress = READ("Email Address:") as STRING

            patientID = zone + group + count
            READ patientID as STRING

            add patientID into patient
            add patientName into patient
            add age into patient
            add group into patient
            add zone into patient
            add contactNumber into patient
            add emailAddress into patient
            PRINT(patient)

            OPEN and APPEND Patient_Detail.txt as fileHandler
            FOR EACH items IN patient
                add items into fileHandler
                add TAB into fileHandler
            ENDFOR
            add new line into fileHandler
            CLOSE Patient_Detail.txt
        ELSE
            PRINT("Patient Registered")
        ENDIF
    ENDDO
ENDFUNCTION
```

```
FUNCTION searchTest(fileName,patientID)
    OPEN and READ fileName as fileHandler
    FOR EACH line IN fileHandler
        IF (patientID IN line) THEN
            PRINT(line)
            CLOSE fileHandler
            RETURN TRUE
        ENDIF
    ENDFOR
    CLOSE fileHandler
    RETURN FALSE
ENDFUNCTION

FUNCTION searchTest1(fileName1,fileName2,patientID)
    OPEN and READ fileName1 as fileHandler
    FOR EACH line IN fileHandler
        IF (patientID IN line) THEN
            PRINT(line)
            CLOSE fileHandler
            RETURN FALSE
        ENDIF
    ENDFOR
    CLOSE fileHandler

    OPEN and READ fileName2 as fileHandler
    FOR EACH line IN fileHandler
        IF (patientID IN line) THEN
            PRINT(line)
            CLOSE fileHandler
            RETURN FALSE
        ENDIF
    ENDFOR
    CLOSE fileHandler
    RETURN TRUE
ENDFUNCTION

FUNCTION testResults()
    choice=0
    DOWHILE (choice not equal to 4)
        PRINT('Select a test')
        PRINT('1. Test 1')
        PRINT('2. Test 2')
        PRINT('3. Test 3')
        PRINT('4. Exit')
        choice=READ('Enter selection: ')
        IF (choice is an integer) THEN
            IF (choice = 1) THEN
                Go to FUNCTION test1()
            ELSEIF (choice = 2) THEN
                Go to FUNCTION test2()
            ELSEIF (choice = 3) THEN
                Go to FUNCTION test3()
            ELSEIF (choice = 4) THEN
                BREAK
            ELSE
                PRINT('Invalid input')
            ENDIF
        ELSE
            PRINT("Non-numeric value entered.")
        ENDIF
    ENDDO
ENDFUNCTION

FUNCTION test1()
    positive=[]
    negative=[]
    WHILE TRUE
        patient=[]
        testNumber='T1'
        patientID=READ("Enter Patient ID,(x) to exit:")in UPPER case
        add patientID into patient
        IF (patientID='X') THEN
            BREAK
        ENDIF
        IF (searchTest1("Test1_negative.txt","Test1_positive.txt",patientID)) THEN
            IF (searchTest("Patient_Detail.txt",patientID)) THEN
                zone=READ("Zone(A/B/C/D):")in UPPER case
                add zone into patient
                IF (zone='A' or zone='B' or zone='C' or zone='D') THEN
                    testResult=READ("Enter Test result,negative or positive:")in LOWER case
                    add testNumber into patient
                    add testResult into patient
                    IF (testResult='negative') THEN
                        group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
                        add group into patient
                        IF (group='AHS') THEN
                            actionTaken='CWFR'
                            PRINT('Continue Working')
                            PRINT("Pls come back for second test.")
                        ELIF (group='SID') THEN
                            actionTaken='HQFR'
                            PRINT('Home Quarantine')
                            PRINT("Pls come back for second test.")
                        ELIF (group='ATO' or group='ACC' or group='AEO') THEN
                            actionTaken='QDFR'
                            PRINT('Quarantine in Designated Centres')
                            PRINT("Pls come back for second test.")
                        ELSE
                            PRINT("Invalid group.")
                            BREAK
                        ENDIF
                        add actionTaken into patient
                        PRINT(patient)
                        add patient into negative

                        OPEN and APPEND Test1_negative.txt as fileHandler
                        FOR EACH items IN negative
                            FOR EACH item IN items
                                add item into fileHandler
                                add TAB into fileHandler
                            ENDFOR
                            add new line into fileHandler
                        ENDFOR
                        CLOSE Test1_negative.txt

                    ELIF (testResult='positive') THEN
                        positivePatientData=[]
                        patientStatus='ACTIVE'
                        group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING

                        add group into patient

                        OPEN and READ Patient_Status.txt as getID
                        count=0
                        FOR EACH line IN getID
                            count=count+1
                        ENDFOR
                        count=count+1
                        CLOSE Patient_Status.txt
                        caseID="C"+zone+group+count
                        READ caseID as STRING

                        add caseID into positivePatientData
                        add patientID into positivePatientData
                        add zone into positivePatientData
                        add group into positivePatientData
                        add patientStatus into positivePatientData

                        IF (group='AHS') THEN
                            actionTaken='HQNF'
                            place="
                            PRINT("Test Result is positive.")
                            PRINT("Home Quarantine")
                        ELIF (group='ATO' or group='ACC' or group='AEO'or group='SID') THEN

                            actionTaken='QHNF'
                            place=READ('WARD or ICU:')in UPPER case
                            IF (place='WARD') THEN
                                PRINT("Test Result is positive.")
                                PRINT('Quarantine in Hospital Normal Ward.')

                            ELIF (place='ICU') THEN
                                PRINT("Test Result is positive.")
                                PRINT('Quarantine in Hospital ICU.')
                            ELSE
                                PRINT("Invalid quarantine place.")
                                BREAK
                            ENDIF
                        ELSE
                            PRINT("Invalid group.")
                            BREAK
                        ENDIF
                        add place into positivePatientData

                        OPEN and APPEND Patient_Status.txt as fileHandler
                        FOR EACH items IN positivePatientData
```

```
                            add item into fileHandler
                            add TAB into fileHandler
                        ENDFOR
                        add new line into fileHandler
                        CLOSE Patient_Status.txt

                        add actionTaken into patient
                        add place into patient
                        PRINT(patient)
                        add patient into positive

                        OPEN and APPEND Test1_positive.txt as fileHandler
                        FOR EACH items IN positive
                            FOR EACH item IN items
                                add item into fileHandler
                                add TAB into fileHandler
                            ENDFOR
                            add new line into fileHandler
                        ENDFOR
                        CLOSE Test1_positive.txt

                    ELSE
                        PRINT("Invalid test result.")
                    ENDIF
                ELSE
                    PRINT("Invalid zone")
                ENDIF
            ELSE
                PRINT("Patient not found.")
            ENDIF
        ELSE
            PRINT("Patient already done test1")
        ENDIF
        BREAK
    ENDWHILE
ENDFUNCTION

FUNCTION test2()
    positive=[]
    negative=[]
    WHILE TRUE
        patient=[]
        testNumber='T2'
        patientID=READ("Enter Patient ID,(x) to exit:")in UPPER case
        add patientID into patient
        IF (patientID='X') THEN
            BREAK
        ENDIF
        IF (searchTest1("Test2_negative.txt","Test2_positive.txt",patientID)) THEN
            IF (searchTest("Test1_negative.txt",patientID)) THEN
                zone=READ("Zone(A/B/C/D):")in UPPER case
                add zone into patient
                IF (zone='A' or zone='B' or zone='C' or zone='D') THEN
                    testResult=READ("Enter Test result,negative or positive:")in LOWER case
                    add testNumber into patient
                    add testResult into patient
                    IF (testResult='negative') THEN
                        group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING

                        add group into patient
                        IF (group='AHS') THEN
                            actionTaken='CWFR'
                            PRINT('Continue Working')
                            PRINT("Pls come back for third test.")
                        ELIF (group='SID') THEN
                            actionTaken='HQFR'
                            PRINT('Home Quarantine')
                            PRINT("Pls come back for third test.")
                        ELIF (group='ATO' or group='ACC' or group='AEO') THEN
                            actionTaken='QDFR'
                            PRINT('Quarantine in Designated Centres')
                            PRINT("Pls come back for third test.")
                        ELSE
                            PRINT("Invalid group.")
                            BREAK
                        ENDIF
                        add actionTaken into patient
                        PRINT(patient)
                        add patient into negative

                        OPEN and APPEND Test2_negative.txt as fileHandler
                        FOR EACH items IN negative
                            FOR EACH item IN items
                                add item into fileHandler
                                add TAB into fileHandler
                            ENDFOR
                            add new line into fileHandler
                        ENDFOR
                        CLOSE Test2_negative.txt

                    ELIF (testResult='positive') THEN
                        positivePatientData=[]
                        patientStatus='ACTIVE'
                        group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING

                        add group into patient

                        OPEN and READ Patient_Status.txt as getID
                        count=0
                        FOR EACH line IN getID
                            count=count+1
                        ENDFOR
                        count=count+1
                        CLOSE Patient_Status.txt
                        caseID="C"+zone+group+count
                        READ caseID as STRING

                        add caseID into positivePatientData
                        add patientID into positivePatientData
                        add zone into positivePatientData
                        add group into positivePatientData
```

```
                add patientStatus into positivePatientData

                IF (group='AHS') THEN
                    actionTaken='HQNF'
                    place=''
                    PRINT("Test Result is positive.")
                    PRINT("Home Quarantine")
                ELIF (group='ATO' or group='ACC' or group='AEO'or
group='SID') THEN
                    actionTaken='QHNF'
                    place=READ('WARD or ICU:')in UPPER case
                    IF (place='WARD') THEN
                        PRINT("Test Result is positive.")
                        PRINT('Quarantine in Hospital Normal
Ward.')

                    ELIF (place='ICU') THEN
                        PRINT("Test Result is positive.")
                        PRINT('Quarantine in Hospital ICU.')
                    ELSE
                        PRINT("Invalid quarantine place.")
                        BREAK
                    ENDIF
                ELSE
                    PRINT("Invalid group.")
                    BREAK
                ENDIF
                add place into positivePatientData

                OPEN and APPEND Patient_Status.txt as fileHandler
                FOR EACH items IN positivePatientData
                    add items into fileHandler
                    add TAB into fileHandler
                ENDFOR
                add new line into fileHandler
                CLOSE Patient_Status.txt

                add actionTaken into patient
                add place into patient
                PRINT(patient)
                add patient into positive

                OPEN and APPEND Test2_positive.txt as fileHandler
                FOR EACH items IN positive
                    FOR EACH item IN items
                        add item into fileHandler
                        add TAB into fileHandler
                    ENDFOR
                    add new line into fileHandler
                ENDFOR
                CLOSE Test2_positive.txt

            ELSE
                PRINT("Invalid test result.")
            ENDIF
        ELSE
            PRINT("Invalid zone")
```

```
                        ENDIF
                    ELSE
                        PRINT('Pls completed test1 first.')
                        PRINT('Positive patients did not need to run test2')
                    ENDIF
                ELSE
                    PRINT("Patient already done test2")
                ENDIF
                BREAK
        ENDWHILE
ENDFUNCTION

FUNCTION test3()
        positive=[]
        negative=[]
        WHILE TRUE
                patient=[]
                testNumber='T3'
                patientID=READ("Enter Patient ID,(x) to exit:")in UPPER case
                add patientID into patient
                IF (patientID='X') THEN
                    BREAK
                ENDIF
                IF (searchTest1("Test3_negative.txt","Test3_positive.txt",patientID)) THEN
                    IF (searchTest("Test2_negative.txt",patientID)) THEN
                        zone=READ("Zone(A/B/C/D):")in UPPER case
                        add zone into patient
                        IF (zone='A' or zone='B' or zone='C' or zone='D') THEN
                            testResult=READ("Enter Test result,negative or positive:")in LOWER
case
                            add testNumber into patient
                            add testResult into patient
                            IF (testResult='negative') THEN
                                group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in
UPPER case as STRING
                                add group into patient
                                IF (group='AHS') THEN
                                    actionTaken='CW'
                                    PRINT('Continue Working')
                                    PRINT("Congratulation!Your last test result is
negative.")
                                ELIF (group='ATO' or group='ACC' or group='AEO' or
group='SID') THEN
                                    actionTaken='RU'
                                    PRINT('Allow to reunion with family.')
                                    PRINT("Congratulation!Your last test result is
negative.")
                                ELSE
                                    PRINT("Invalid group.")
                                    BREAK
                                ENDIF
                                add actionTaken into patient
                                PRINT(patient)
                                add patient into negative

                                OPEN and APPEND Test3_negative.txt as fileHandler
```

```
    FOR EACH items IN negative
        FOR EACH item IN items
            add item into fileHandler
            add TAB into fileHandler
        ENDFOR
        add new line into fileHandler
    ENDFOR
    CLOSE Test3_negative.txt

ELIF (testResult='positive') THEN
    positivePatientData=[]
    patientStatus='ACTIVE'
    group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING

    add group into patient

    OPEN and READ Patient_Status.txt as getID
    count=0
    FOR EACH line IN getID
        count=count+1
    ENDFOR
    count=count+1
    CLOSE Patient_Status.txt
    caseID="C"+zone+group+count
    READ caseID as STRING

    add caseID into positivePatientData
    add patientID into positivePatientData
    add zone into positivePatientData
    add group into positivePatientData
    add patientStatus into positivePatientData

    IF (group='AHS') THEN
        actionTaken='HQNF'
        place="
        PRINT("Test Result is positive.")
        PRINT("Home Quarantine")
    ELIF (group='ATO' or group='ACC' or group='AEO'or group='SID') THEN
        actionTaken='QHNF'
        place=READ('WARD or ICU:')in UPPER case
        IF (place='WARD') THEN
            PRINT("Test Result is positive.")
            PRINT('Quarantine in Hospital Normal Ward.')

        ELIF (place='ICU') THEN
            PRINT("Test Result is positive.")
            PRINT('Quarantine in Hospital ICU.')
        ELSE
            PRINT("Invalid quarantine place.")
            BREAK
        ENDIF
    ELSE
        PRINT("Invalid group.")
        BREAK
    ENDIF
```

```
                        add place into positivePatientData

                        OPEN and APPEND Patient_Status.txt as fileHandler
                        FOR EACH items IN positivePatientData
                            add item into fileHandler
                            add TAB into fileHandler
                        ENDFOR
                        add new line into fileHandler
                        CLOSE Patient_Status.txt

                        add actionTaken into patient
                        add place into patient
                        PRINT(patient)
                        add patient into positive

                        OPEN and APPEND Test3_positive.txt as fileHandler
                        FOR EACH items IN positive
                            FOR EACH item IN items
                                add item into fileHandler
                                add TAB into fileHandler
                            ENDFOR
                            add new line into fileHandler
                        ENDFOR
                        CLOSE Test3_positive.txt

                    ELSE
                        PRINT("Invalid test result.")
                    ENDIF
                ELSE
                    PRINT("Invalid zone")
                ENDIF
            ELSE
                PRINT('Pls completed test2 first.')
                PRINT('Positive patients did not need to run test3')
            ENDIF
        ELSE
            PRINT("Patient already done test3")
        ENDIF
        BREAK
    ENDWHILE
ENDFUNCTION
```

```
FUNCTION modifyPatientStatus()
    OPEN and READ Patient_Status.txt as fileHandler
    caseID=READ("Enter Patient's Case ID,(x) to exit: ") in UPPER case
    fileData=READLINES(fileHandler)
    FOR EACH index,line IN ENUMERATE(fileData)
        IF (caseID IN line) THEN
            PRINT(line)
            patientStatus=READ("Patient Status(ACTIVE/RECOVERED/DECEASED):") in
UPPER case
            REPLACE "ACTIVE" in line with patientStatus
            fileData[index]=line
            PRINT(line)
        ENDIF
    ENDFOR
    CLOSE Patient_Status.txt

    OPEN and WRITE Patient_Status.txt as file
    FOR EACH line IN fileData
        add line into file
    ENDFOR
    CLOSE Patient_Status.txt
ENDFUNCTION
```

```
FUNCTION testCarriedOut()
    count1=0
    count2=0
    count3=0
    OPEN and READ Test1_negative.txt as fileHandler
    FOR EACH items IN fileHandler
        count1=count1+1
    ENDFOR
    CLOSE Test1_negative.txt

    OPEN and READ Test1_positive.txt as fileHandler
    FOR EACH items IN fileHandler
        count1=count1+1
    ENDFOR
    CLOSE Test1_positive.txt

    OPEN and READ Test2_negative.txt as fileHandler
    FOR EACH items IN fileHandler
        count2=count2+1
    ENDFOR
    CLOSE Test2_negative.txt

    OPEN and READ Test2_positive.txt as fileHandler
    FOR EACH items IN fileHandler
        count2=count2+1
    ENDFOR
    CLOSE Test2_positive.txt

    OPEN and READ Test3_negative.txt as fileHandler
    FOR EACH items IN fileHandler
        count3=count3+1
    ENDFOR
    CLOSE Test3_negative.txt

    OPEN and READ Test3_positive.txt as fileHandler
    FOR EACH items IN fileHandler
        count3=count3+1
    ENDFOR
    CLOSE Test3_positive.txt

    PRINT("Total number of Test1 carried out is",count1)
    PRINT("Total number of Test2 carried out is",count2)
    PRINT("Total number of Test3 carried out is",count3)
```

```
ENDFUNCTION

FUNCTION patientsTested()
    count=0
    OPEN and READ Test1_negative.txt as fileHandler
    FOR EACH items IN fileHandler
        count=count+1
    ENDFOR
    CLOSE Test1_negative.txt

    OPEN and READ Test1_positive.txt as fileHandler
    FOR EACH items IN fileHandler
        count=count+1
    ENDFOR
    CLOSE Test1_positive.txt
    PRINT("Total number of patients tested is",count)
ENDFUNCTION

FUNCTION recoveredCases()
    count=0
    status="RECOVERED"
    OPEN and READ Patient_Status.txt as fileHandler
    FOR EACH items IN fileHandler
        IF (status IN items) THEN
            count=count+1
        ENDIF
    ENDFOR
    CLOSE Patient_Status.txt
    PRINT("Total number of recovered cases is",count)
ENDFUNCTION

FUNCTION positiveGroup()
    count1=0
    count2=0
    count3=0
    count4=0
    count5=0
    OPEN and READ Test1_positive.txt as fileHandler
    FOR EACH item IN fileHandler
        IF (item[1to4]='ATO') THEN
            count1=count1+1
        ELSEIF (item[1to4]='ACC') THEN
            count2=count2+1
        ELSEIF (item[1to4]='AEO') THEN
            count3=count3+1
        ELSEIF (item[1to4]='SID') THEN
            count4=count4+1
        ELSEIF (item[1to4]='AHS') THEN
            count5=count5+1
        ENDIF
    ENDFOR
    CLOSE Test1_positive.txt

    OPEN and READ Test2_positive.txt as fileHandler
    FOR EACH item IN fileHandler
        IF item[1to4]='ATO' THEN
```

```
                count1=count1+1
            ELSEIF item[1to4]='ACC' THEN
                count2=count2+1
            ELSEIF item[1to4]='AEO' THEN
                count3=count3+1
            ELSEIF item[1to4]='SID' THEN
                count4=count4+1
            ELSEIF item[1to4]='AHS' THEN
                count5=count5+1
            ENDIF
        ENDFOR
        CLOSE Test2_positive.txt

        OPEN and READ Test3_positive.txt as fileHandler
        FOR EACH item IN fileHandler
            IF (item[1to4]='ATO') THEN
                count1=count1+1
            ELSEIF (item[1to4]='ACC') THEN
                count2=count2+1
            ELSEIF (item[1to4]='AEO') THEN
                count3=count3+1
            ELSEIF (item[1to4]='SID') THEN
                count4=count4+1
            ELSEIF (item[1to4]='AHS') THEN
                count5=count5+1
            ENDIF
        ENDFOR
        CLOSE Test3_positive.txt
        PRINT("Total number of positive patients in ATO",count1)
        PRINT("Total number of positive patients in ACC",count2)
        PRINT("Total number of positive patients in AEO",count3)
        PRINT("Total number of positive patients in SID",count4)
        PRINT("Total number of positive patients in AHS",count5)
ENDFUNCTION

FUNCTION positiveZone()
        count1=0
        count2=0
        count3=0
        count4=0
        OPEN AND READ Test1_positive.txt as fileHandler
        FOR EACH item IN fileHandler
            IF (item[0]='A') THEN
                count1=count1+1
            ELSEIF (item[0]='B') THEN
                count2=count2+1
            ELSEIF (item[0]='C') THEN
                count3=count3+1
            ELSEIF (item[0]='D') THEN
                count4=count4+1
            ENDIF
        ENDFOR
        CLOSE Test1_positive.txt

        OPEN AND READ Test2_positive.txt as fileHandler
        FOR EACH item IN fileHandler
            IF (item[0]='A') THEN
                count1=count1+1
            ELSEIF (item[0]='B') THEN
                count2=count2+1
            ELSEIF (item[0]='C') THEN
                count3=count3+1
            ELSEIF (item[0]='D') THEN
                count4=count4+1
            ENDIF
        ENDFOR
        CLOSE Test2_positive.txt

        OPEN AND READ Test3_positive.txt as fileHandler
        FOR EACH item IN fileHandler
            IF (item[0]='A') THEN
                count1=count1+1
            ELSEIF (item[0]='B') THEN
                count2=count2+1
            ELSEIF (item[0]='C') THEN
                count3=count3+1
            ELSEIF (item[0]='D') THEN
                count4=count4+1
            ENDIF
        ENDFOR
        CLOSE Test3_positive.txt
        PRINT("Total number of positive patients in Zone A",count1)
        PRINT("Total number of positive patients in Zone B",count2)
        PRINT("Total number of positive patients in Zone C",count3)
        PRINT("Total number of positive patients in Zone D",count4)
ENDFUNCTION

FUNCTION statisticalInformation()
        choice=0
        WHILE choice not equal to 6
            PRINT("Total number of")
            PRINT("1. Test Carried Out")
            PRINT("2. Patients tested")
            PRINT("3. Recovered cases")
            PRINT("4. Patient Test Positive for COVID-19 group wise")
            PRINT("5. Active cases zone wise")
            PRINT("6. Exit")
            PRINT("Enter selection: ")
            READ choice
            IF (choice is an integer) THEN
                IF (choice=1) THEN
                    testCarriedOut()
                ELSEIF (choice=2) THEN
                    patientsTested()
                ELSEIF (choice=3) THEN
                    recoveredCases()
                ELSEIF (choice=4) THEN
                    positiveGroup()
                ELSEIF (choice=5) THEN
                    positiveZone()
                ELSEIF (choice=6) THEN
                    BREAK
                ELSE
                    PRINT('Invalid input.')
                ENDIF
            ELSE
                PRINT("Non-numeric value entered.")
            ENDIF
        ENDWHILE
ENDFUNCTION
```

```
FUNCTION searchPatientRecord()
    OPEN and READ Patient_Detail.txt as fileHandler
    search_key = READ('Enter patient ID or name: ') as UPPER case
    FOR EACH line IN fileHandler
        IF (search_key IN line) THEN
            PRINT(line)
            RETURN
        ENDIF
    ENDFOR
    CLOSE Patient_Detail.txt
ENDFUNCTION

FUNCTION searchCaseStatus()
    OPEN and READ Patient_Status.txt as fileHandler
    search_key = READ('Enter case ID: ') as UPPER case
    FOR EACH line IN fileHandler
        IF (search_key IN line) THEN
            PRINT(line)
            RETURN
        ENDIF
    ENDFOR
    CLOSE Patient_Status.txt
ENDFUNCTION

FUNCTION deceasedPatient()
    OPEN and READ Patient_Status.txt as fileHandler
    search_key = 'DECEASED'
    FOR EACH line IN fileHandler
        IF (search_key IN line) THEN
            PRINT(line)
        ENDIF
    ENDFOR
    CLOSE Patient_Status.txt
ENDFUNCTION

FUNCTION searchPatientData()
    choice=0
    WHILE choice not equal to 4
        PRINT("1. Patient Record")
        PRINT("2. Status of Case")
        PRINT("3. Patient Record of all Decreased Patients")
        PRINT("4. Exit")
        PRINT("Enter selection: ")
        READ choice
        IF (choice is an integer) THEN
```

```
                IF (choice = 1) THEN
                    Go to FUNCTION searhPatientRecord()
                ELSEIF (choice = 2) THEN
                    Go to FUNCTION searchCaseStatus()
                ELSEIF (choice = 3) THEN
                    Go to FUNCTION deceasedPatient()
                ELSEIF (choice = 4) THEN
                    BREAK
                ELSE
                    PRINT('Invalid input')
                ENDIF
            ELSE
                PRINT("Non-numeric value entered.")        \
            ENDIF
    ENDWHILE
ENDFUNCTION

----------------------------------------------------------------------------------------------------------------------------------

FUNCTION createFile()
    OPEN and CLOSE Patient_Detail.txt
    OPEN and CLOSE Test1_negative.txt
    OPEN and CLOSE Test1_positive.txt
    OPEN and CLOSE Test2_negative.txt
    OPEN and CLOSE Test2_positive.txt
    OPEN and CLOSE Test3_negative.txt
    OPEN and CLOSE Test3_positive.txt
    OPEN and CLOSE Patient_Status.txt
ENDFUNCTION

FUNCTION MENU()
    choice=0
    WHILE choice not equal to 6
        PRINT('----------COVID-19 Patient Management System----------')
        PRINT("Select the operation that you want to perform.")
        PRINT("1. New Patient Registration")
        PRINT("2. Test Result and Action Taken")
        PRINT("3. Changing Patient Status")
        PRINT("4. Statistical Information on Tests Carried Out")
        PRINT("5. Searching Functionalities")
        PRINT("6. Exit")
        choice = READ("Enter selection: ")
        createFile()
        IF (choice is an integer) THEN
            IF (choice = 1) THEN
                Go to FUNCTION patientRegistration()
            ELSEIF (choice = 2) THEN
                Go to FUNCTION testResults()
            ELSEIF (choice = 3) THEN
                Go to FUNCTION modifyPatientStatus()
            ELSEIF (choice = 4) THEN
                Go to FUNCTION statisticalInformation()
            ELSEIF (choice = 5) THEN
                Go to FUNCTION searchPatientData()
            ELSEIF (choice = 6) THEN
                BREAK
            ELSE
                PRINT('Invalid input')
            ENDIF
        ELSE
            PRINT("Non-numeric value entered.")
        ENDIF
    ENDWHILE
ENDFUNCTION

MENU()

END
```

2.2 Flowcharts

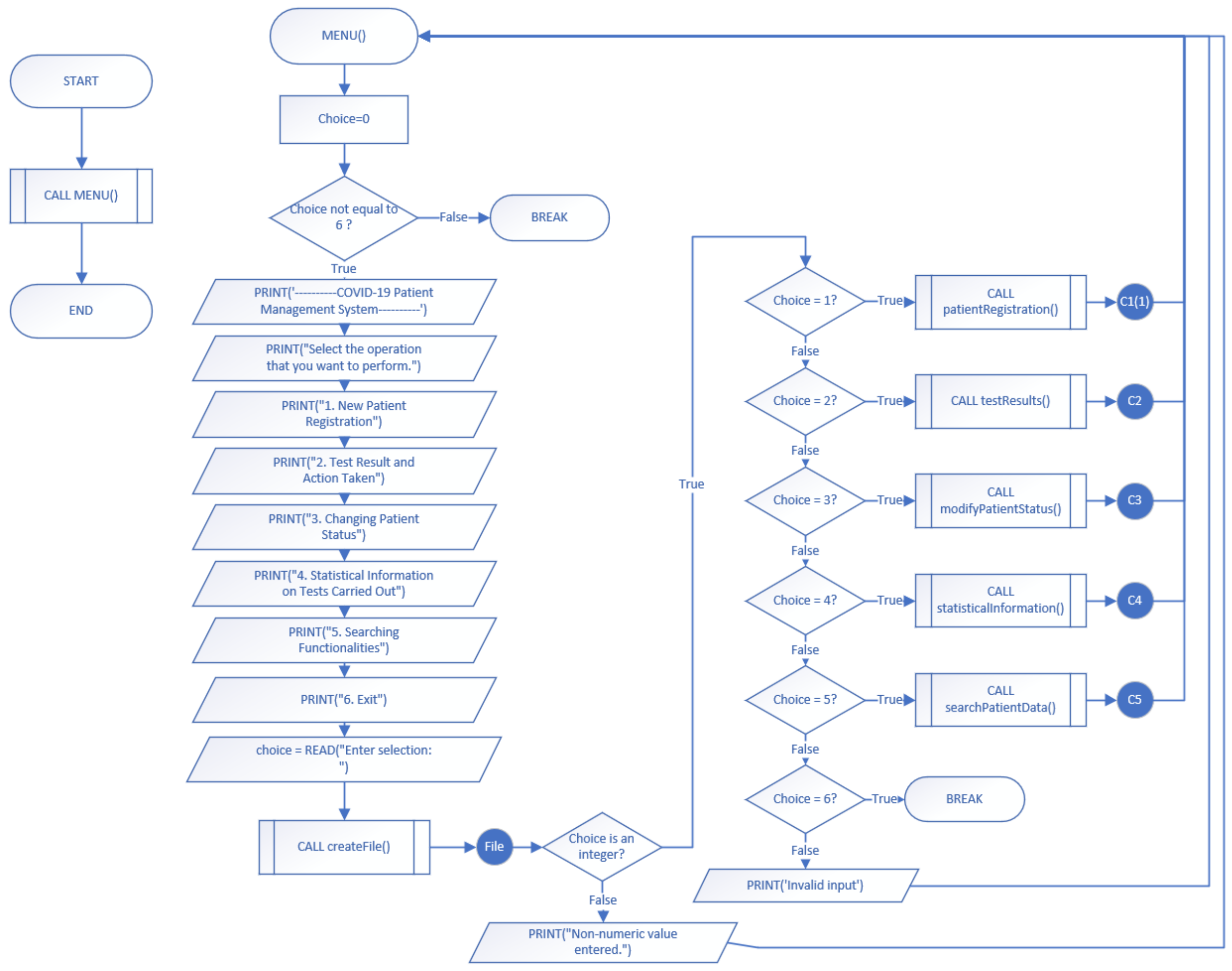

START
CALL MENU()
END
MENU()
Choice=0
Choice not equal to 6 ?
False
BREAK
True
PRINT('----------COVID-19 Patient Management System----------')
PRINT("Select the operation that you want to perform.")
PRINT("1. New Patient Registration")
PRINT("2. Test Result and Action Taken")
PRINT("3. Changing Patient Status")
PRINT("4. Statistical Information on Tests Carried Out")
PRINT("5. Searching Functionalities")
PRINT("6. Exit")
choice = READ("Enter selection: ")
CALL createFile()
File
Choice is an integer?
False
PRINT("Non-numeric value entered.")
True
Choice = 1?
CALL patientRegistration()
C1(1)
Choice = 2?
CALL testResults()
C2
Choice = 3?
CALL modifyPatientStatus()
C3
Choice = 4?
CALL statisticalInformation()
C4
Choice = 5?
CALL searchPatientData()
C5
Choice = 6?
BREAK
PRINT('Invalid input')

Figure2.2.1 Menu

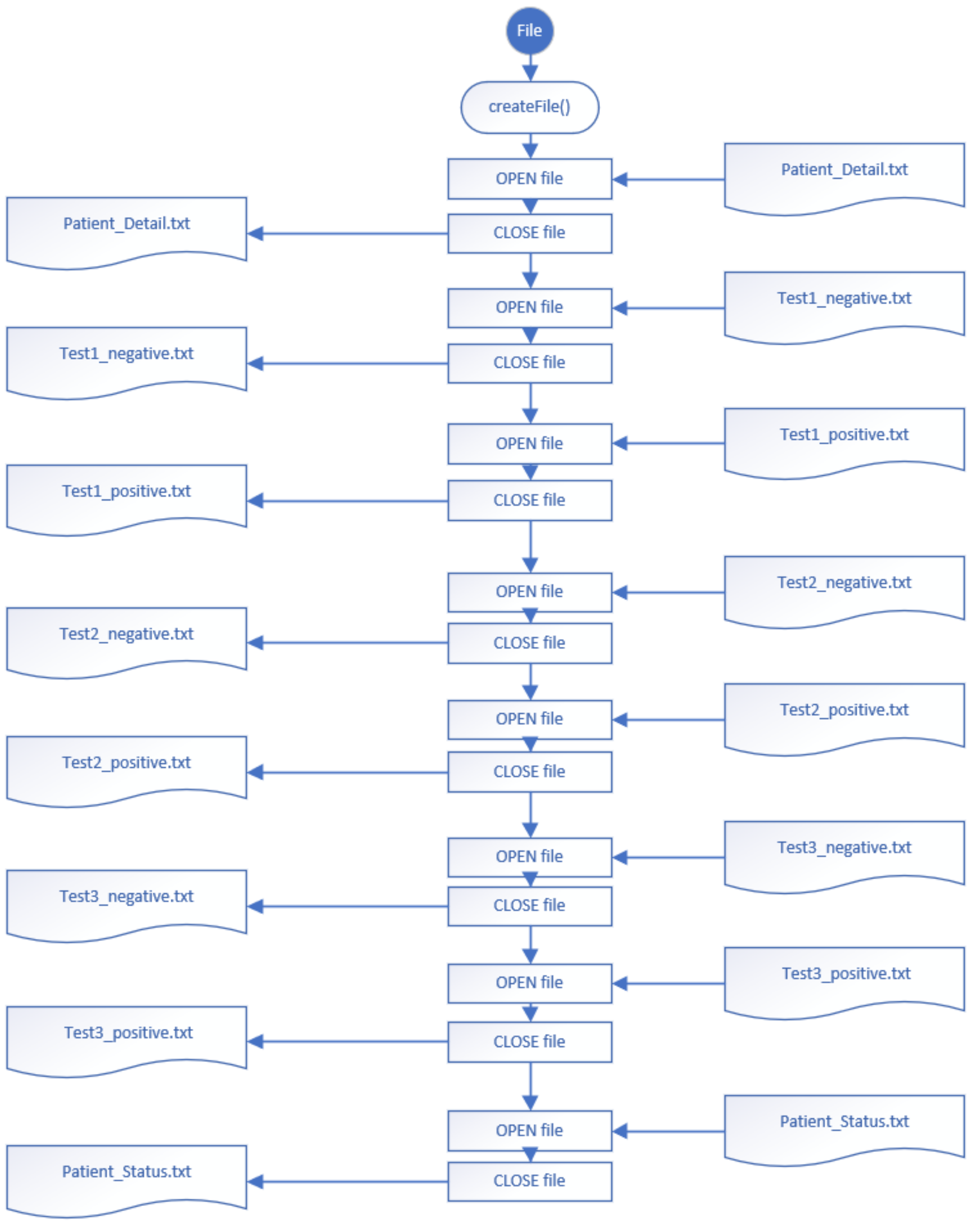


Figure 2.2.2 Create File

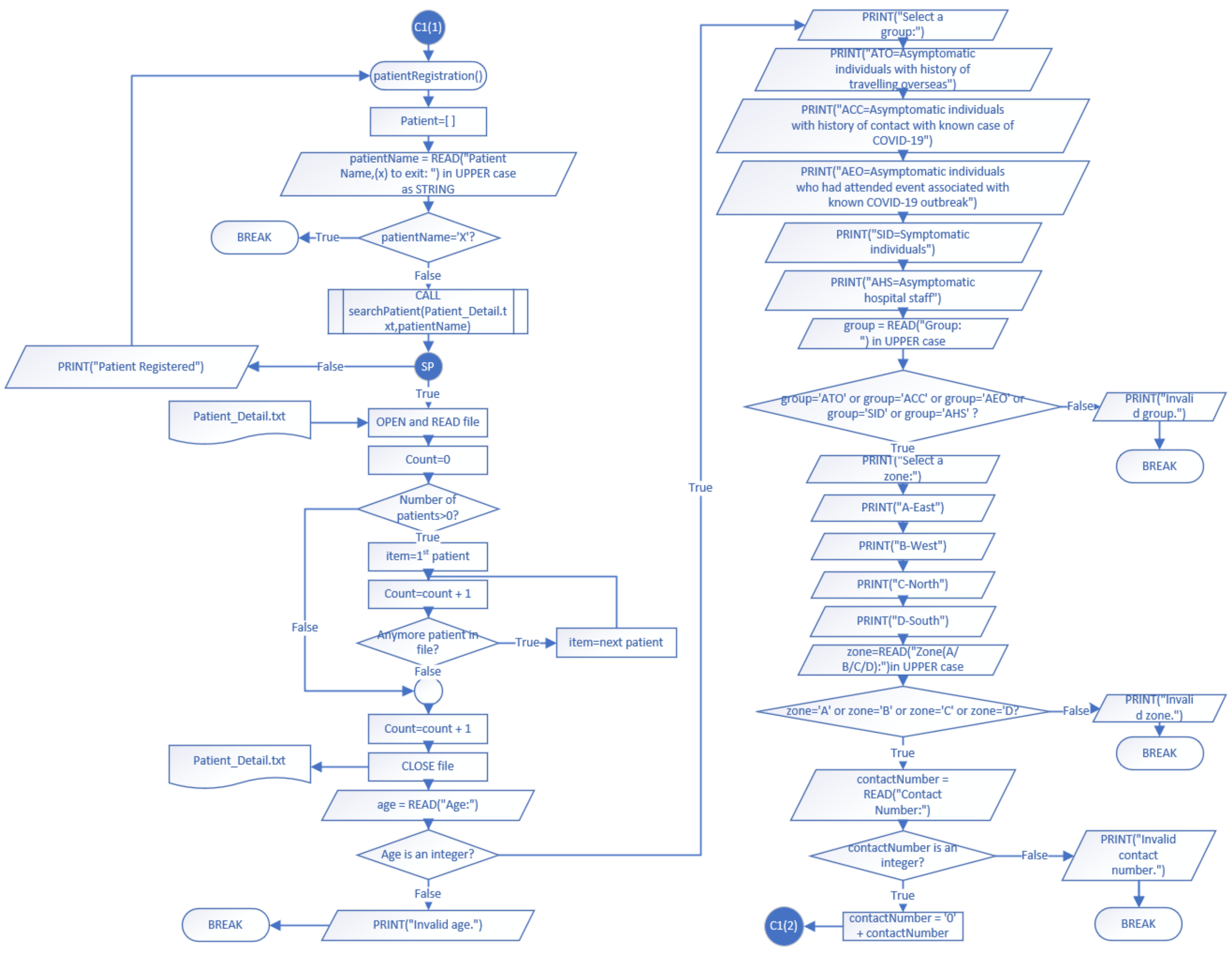

C1(1)
patientRegistration()
Patient=[ ]
patientName = READ("Patient Name,(x) to exit: ") in UPPER case as STRING
patientName='X'?
True
BREAK
False
CALL searchPatient(Patient_Detail.txt,patientName)
SP
False
PRINT("Patient Registered")
True
Patient_Detail.txt
OPEN and READ file
Count=0
Number of patients>0?
True
item=1st patient
Count=count + 1
Anymore patient in file?
True
item=next patient
False
False
Count=count + 1
CLOSE file
Patient_Detail.txt
age = READ("Age:")
Age is an integer?
False
PRINT("Invalid age.")
BREAK
True
PRINT("Select a group:")
PRINT("ATO=Asymptomatic individuals with history of travelling overseas")
PRINT("ACC=Asymptomatic individuals with history of contact with known case of COVID-19")
PRINT("AEO=Asymptomatic individuals who had attended event associated with known COVID-19 outbreak")
PRINT("SID=Symptomatic individuals")
PRINT("AHS=Asymptomatic hospital staff")
group = READ("Group: ") in UPPER case
group='ATO' or group='ACC' or group='AEO' or group='SID' or group='AHS' ?
False
PRINT("Invalid group.")
BREAK
True
PRINT("Select a zone:")
PRINT("A-East")
PRINT("B-West")
PRINT("C-North")
PRINT("D-South")
zone=READ("Zone(A/B/C/D):")in UPPER case
zone='A' or zone='B' or zone='C' or zone='D'?
False
PRINT("Invalid zone.")
BREAK
True
contactNumber = READ("Contact Number:")
contactNumber is an integer?
False
PRINT("Invalid contact number.")
BREAK
True
contactNumber = '0' + contactNumber
C1(2)

Figure 2.2.3 Patient Registration (part 1)

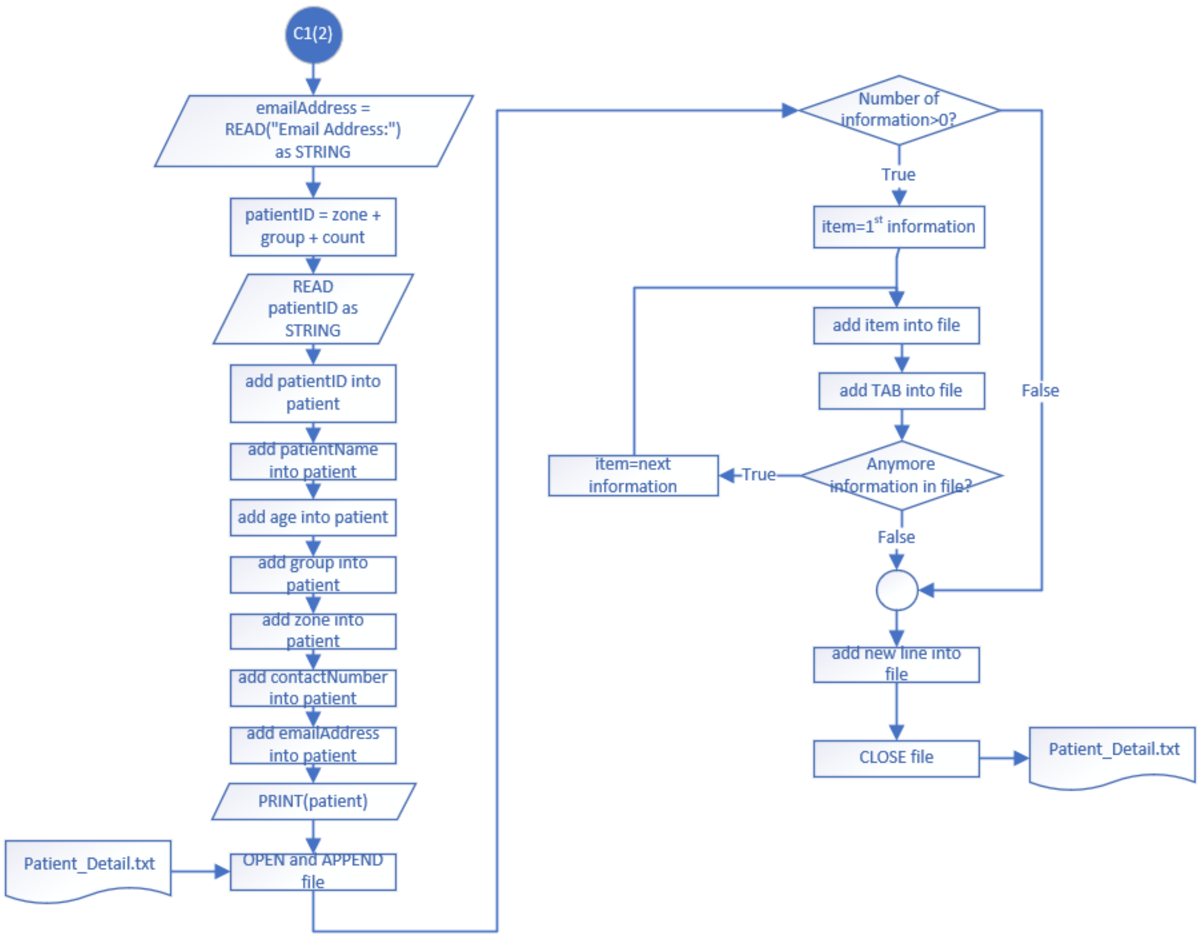


Figure 2.2.4 Patient Registration (part 2)

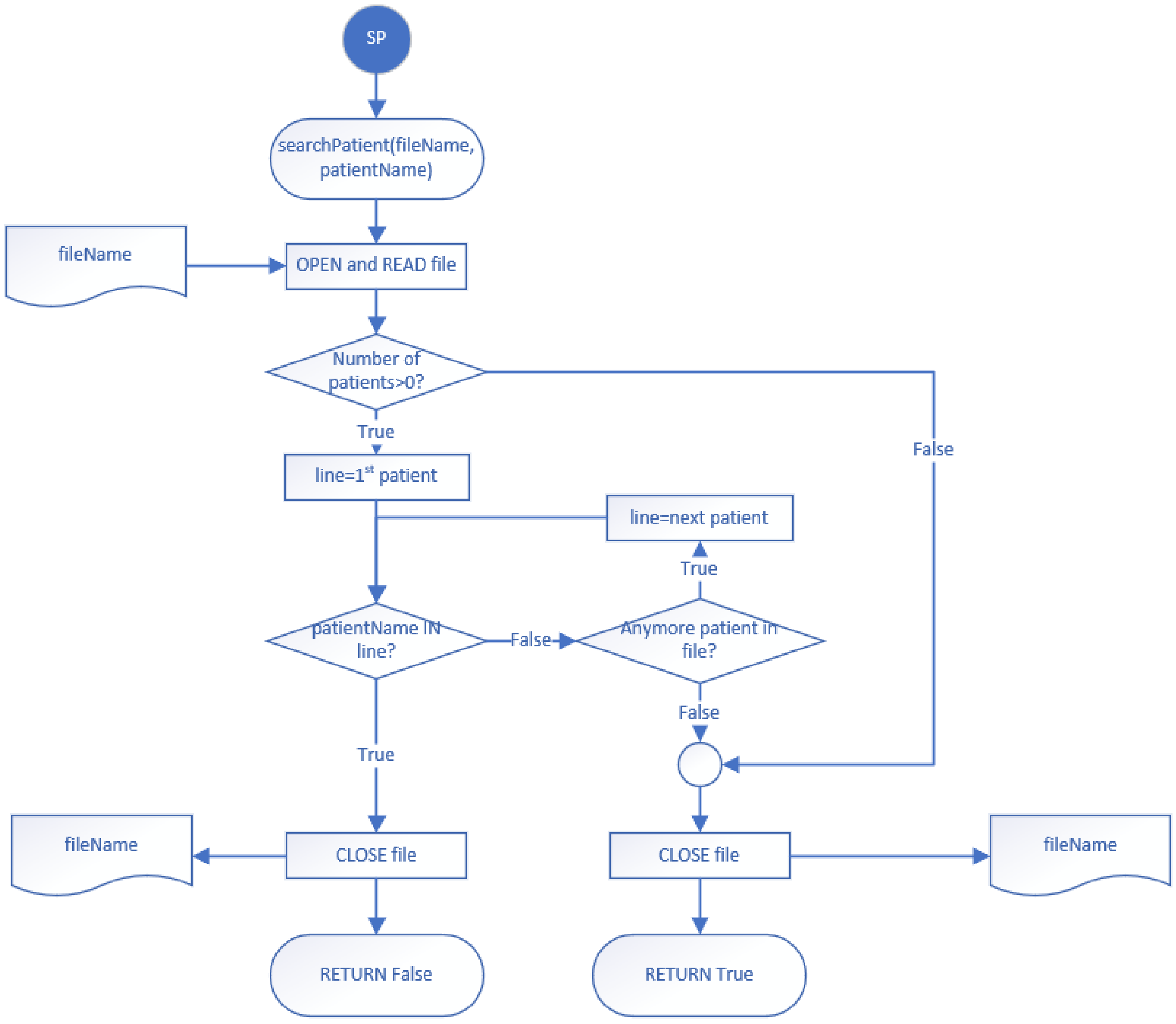


Figure 2.2.5 Search Patient

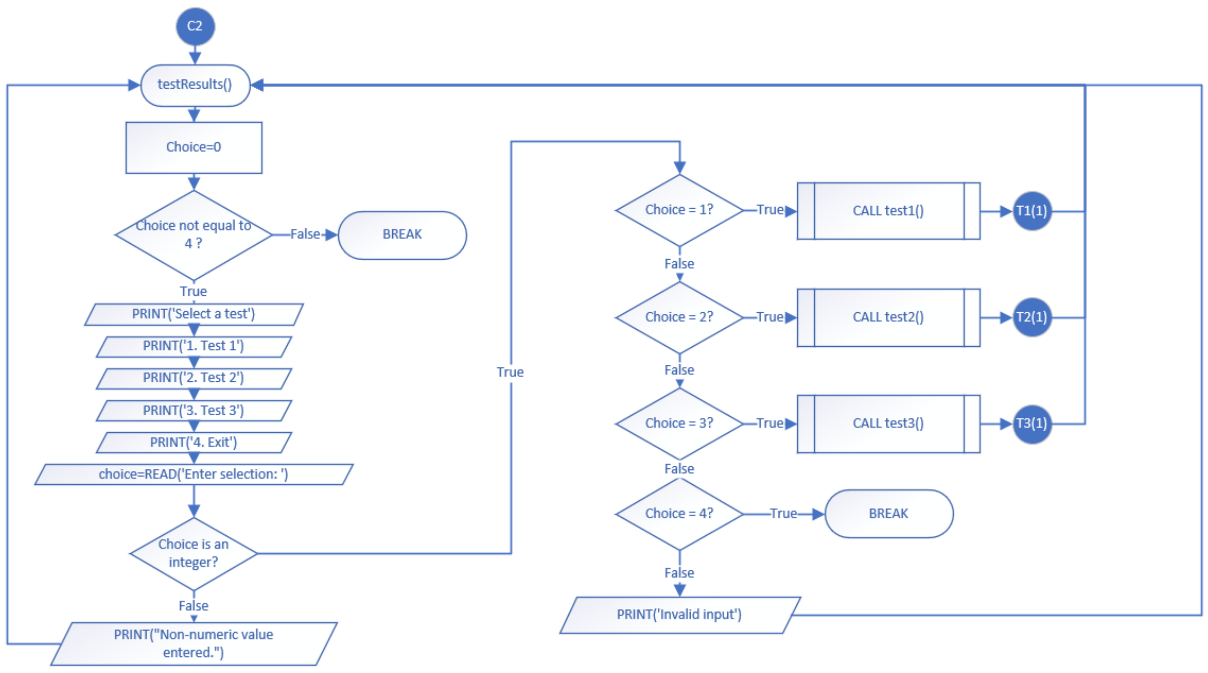


Figure 2.2.6 Test Results

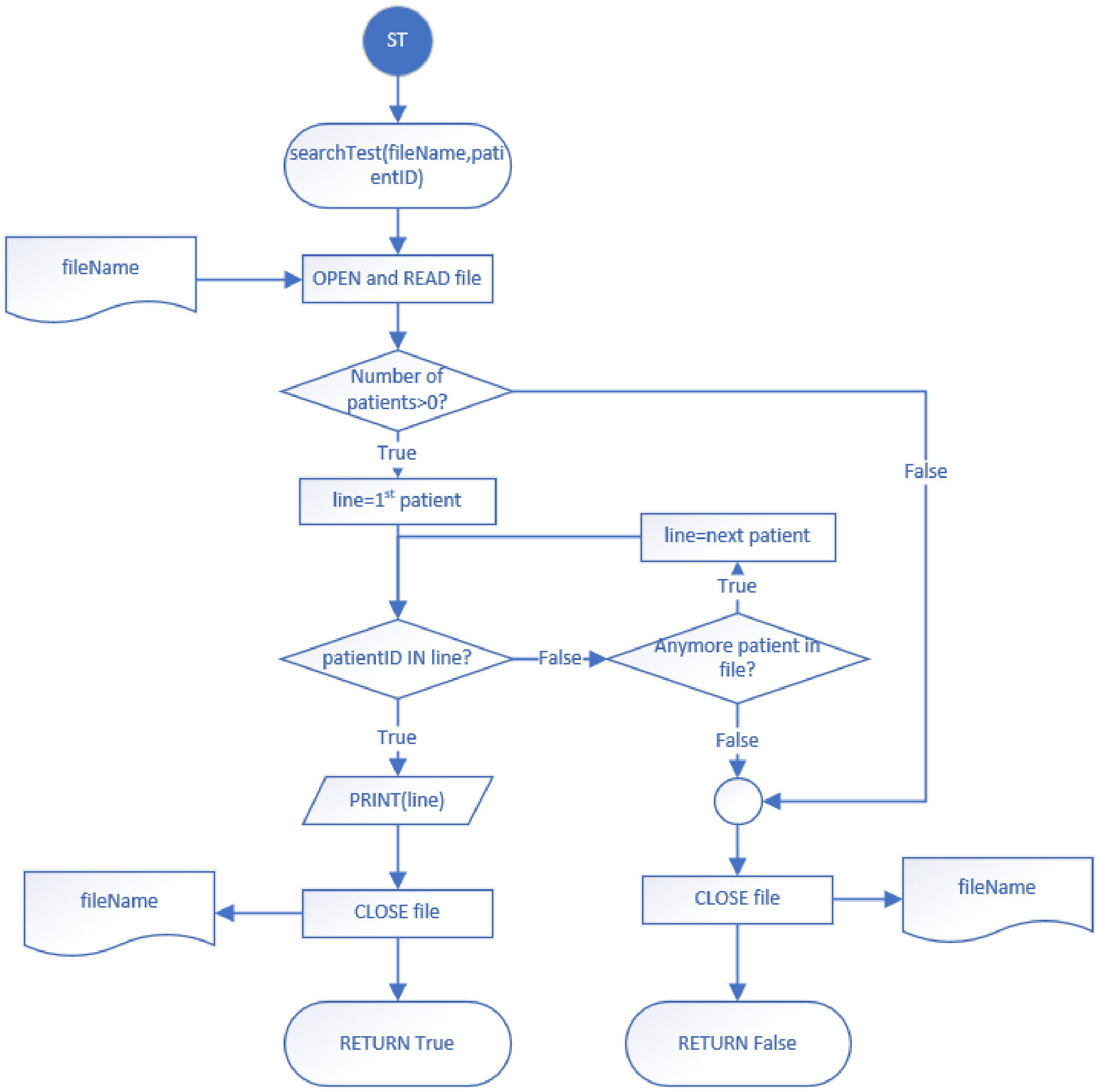


Figure 2.2.7 Search Test

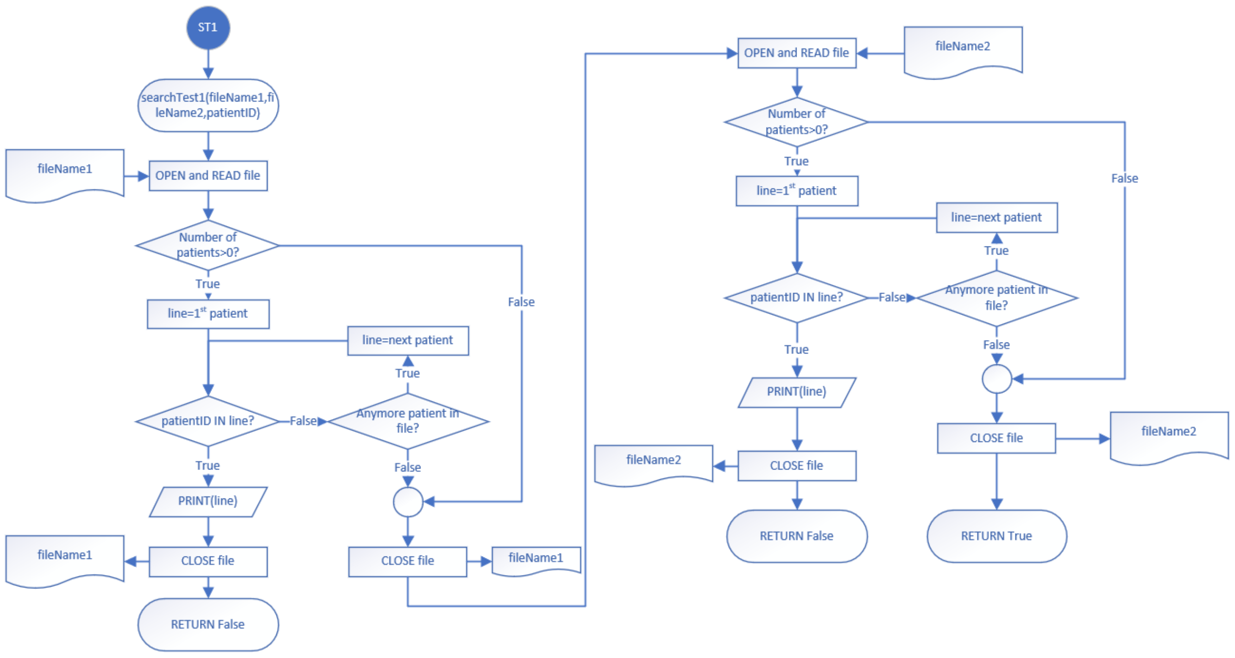


Figure 2.2.8 Search Test1

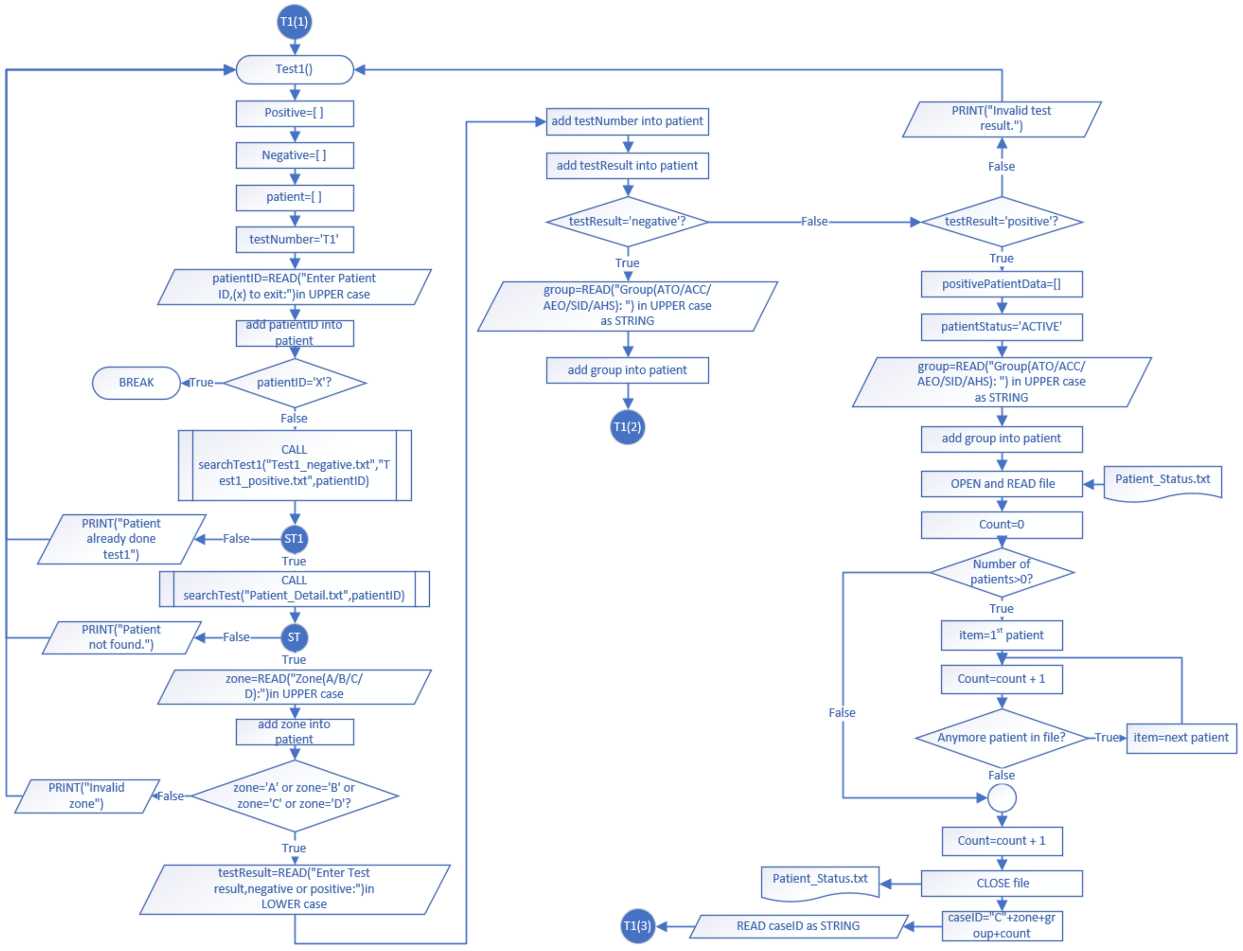

T1(1)
Test1()
Positive=[ ]
Negative=[ ]
patient=[ ]
testNumber='T1'
patientID=READ("Enter Patient ID,(x) to exit:")in UPPER case
add patientID into patient
BREAK
True
patientID='X'?
False
CALL searchTest1("Test1_negative.txt","Test1_positive.txt",patientID)
PRINT("Patient already done test1")
False
ST1
True
CALL searchTest("Patient_Detail.txt",patientID)
PRINT("Patient not found.")
False
ST
True
zone=READ("Zone(A/B/C/D):")in UPPER case
add zone into patient
PRINT("Invalid zone")
False
zone='A' or zone='B' or zone='C' or zone='D'?
True
testResult=READ("Enter Test result,negative or positive:")in LOWER case
add testNumber into patient
add testResult into patient
testResult='negative'?
False
True
group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
add group into patient
T1(2)
PRINT("Invalid test result.")
False
testResult='positive'?
True
positivePatientData=[]
patientStatus='ACTIVE'
group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
add group into patient
OPEN and READ file
Patient_Status.txt
Count=0
Number of patients>0?
True
False
item=1st patient
Count=count + 1
Anymore patient in file?
True
item=next patient
False
Count=count + 1
Patient_Status.txt
CLOSE file
T1(3)
READ caseID as STRING
caseID="C"+zone+group+count

Figure 2.2.9 Test1 (Part1)

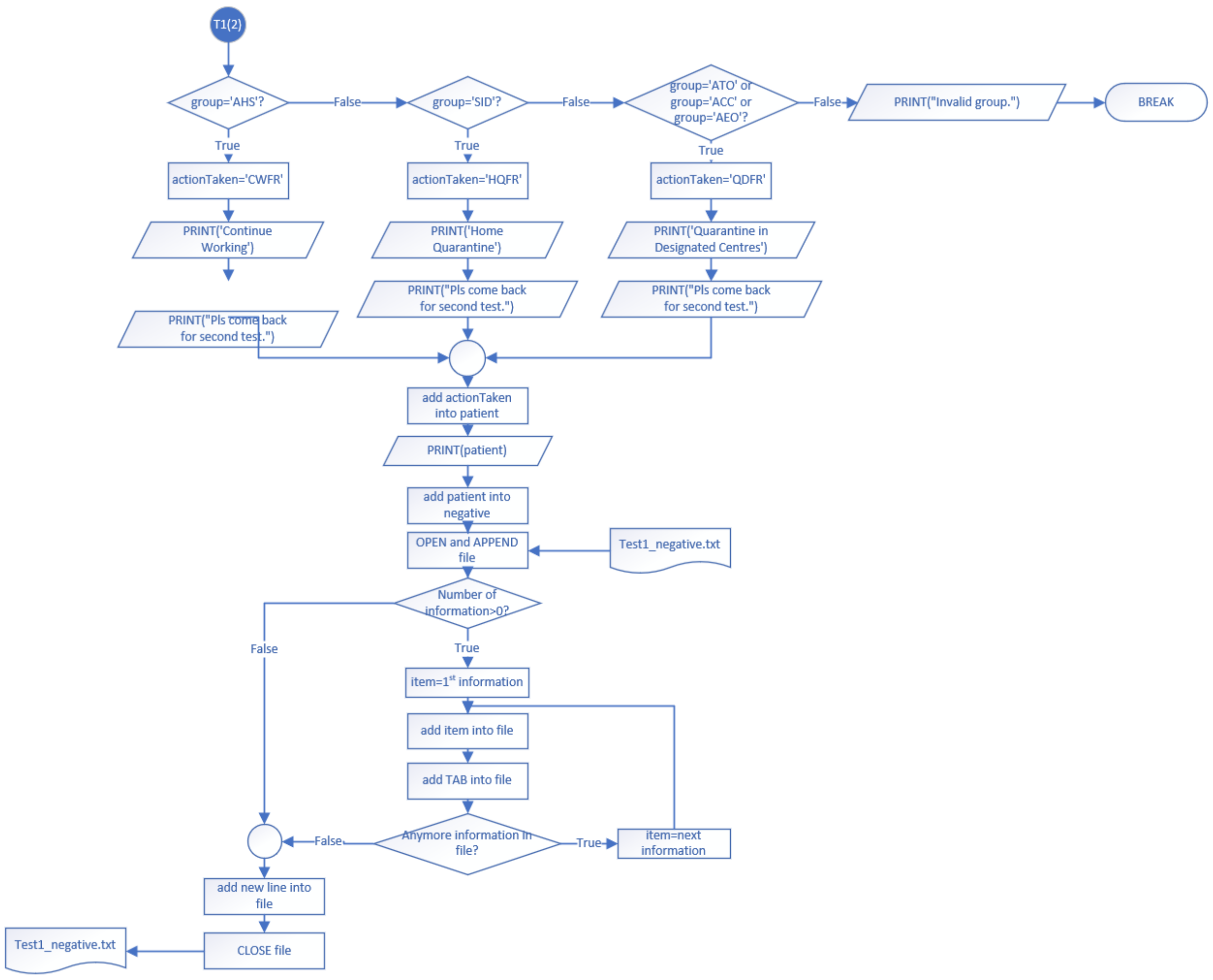


Figure 2.2.10 Test1 (Part2)

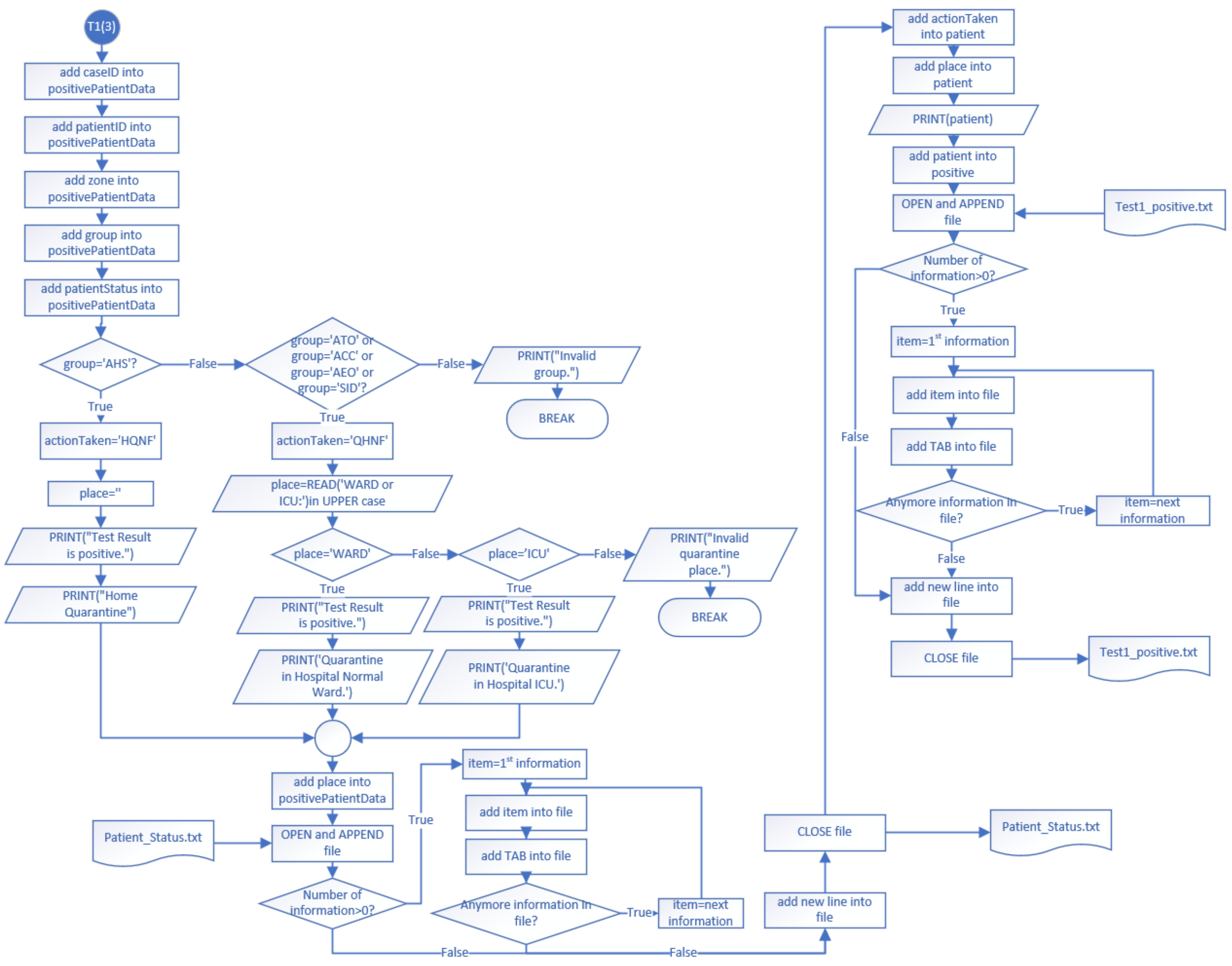

T1(3)
add caseID into positivePatientData
add patientID into positivePatientData
add zone into positivePatientData
add group into positivePatientData
add patientStatus into positivePatientData
group='AHS'?
True
False
actionTaken='HQNF'
place=''
PRINT("Test Result is positive.")
PRINT("Home Quarantine")
group='ATO' or group='ACC' or group='AEO' or group='SID'?
True
False
PRINT("Invalid group.")
BREAK
actionTaken='QHNF'
place=READ('WARD or ICU:')in UPPER case
place='WARD'
True
False
PRINT("Test Result is positive.")
PRINT('Quarantine in Hospital Normal Ward.')
place='ICU'
True
False
PRINT("Test Result is positive.")
PRINT('Quarantine in Hospital ICU.')
PRINT("Invalid quarantine place.")
BREAK
add place into positivePatientData
Patient_Status.txt
OPEN and APPEND file
Number of information>0?
True
False
item=1st information
add item into file
add TAB into file
Anymore information in file?
True
False
item=next information
add new line into file
CLOSE file
Patient_Status.txt
add actionTaken into patient
add place into patient
PRINT(patient)
add patient into positive
OPEN and APPEND file
Test1_positive.txt
Number of information>0?
True
False
item=1st information
add item into file
add TAB into file
Anymore information in file?
True
False
item=next information
add new line into file
CLOSE file
Test1_positive.txt

Figure 2.2.11 Test1 (Part3)

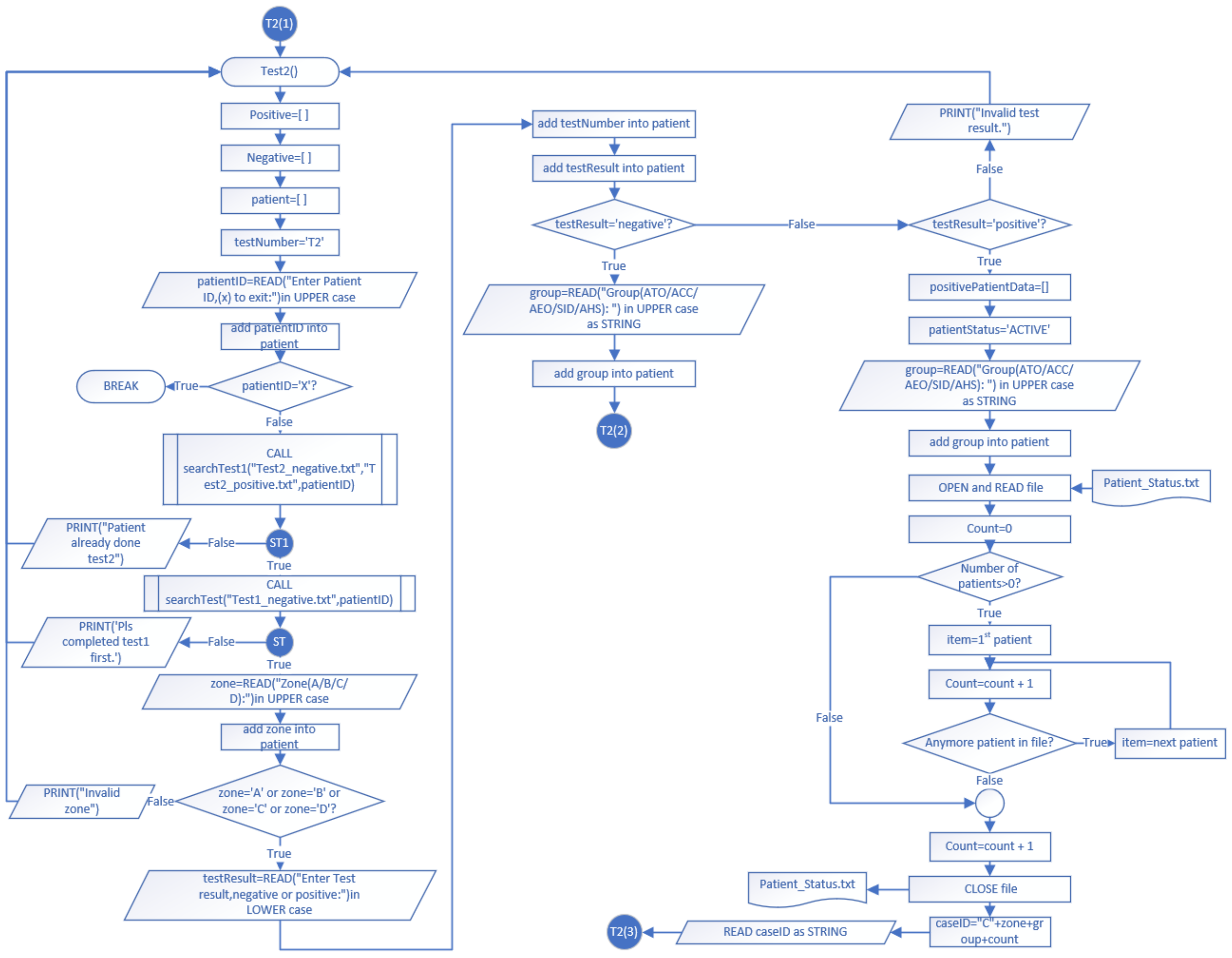

T2(1)
Test2()
Positive=[ ]
Negative=[ ]
patient=[ ]
testNumber='T2'
patientID=READ("Enter Patient ID,(x) to exit:")in UPPER case
add patientID into patient
BREAK
True
patientID='X'?
False
CALL searchTest1("Test2_negative.txt","Test2_positive.txt",patientID)
PRINT("Patient already done test2")
False
ST1
True
CALL searchTest("Test1_negative.txt",patientID)
PRINT('Pls completed test1 first.')
False
ST
True
zone=READ("Zone(A/B/C/D):")in UPPER case
add zone into patient
PRINT("Invalid zone")
False
zone='A' or zone='B' or zone='C' or zone='D'?
True
testResult=READ("Enter Test result,negative or positive:")in LOWER case
add testNumber into patient
add testResult into patient
testResult='negative'?
False
True
group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
add group into patient
T2(2)
PRINT("Invalid test result.")
False
testResult='positive'?
True
positivePatientData=[]
patientStatus='ACTIVE'
group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
add group into patient
OPEN and READ file
Patient_Status.txt
Count=0
Number of patients>0?
True
False
item=1st patient
Count=count + 1
Anymore patient in file?
True
item=next patient
False
Count=count + 1
Patient_Status.txt
CLOSE file
T2(3)
READ caseID as STRING
caseID="C"+zone+group+count

Figure 2.2.12 Test2 (Part1)

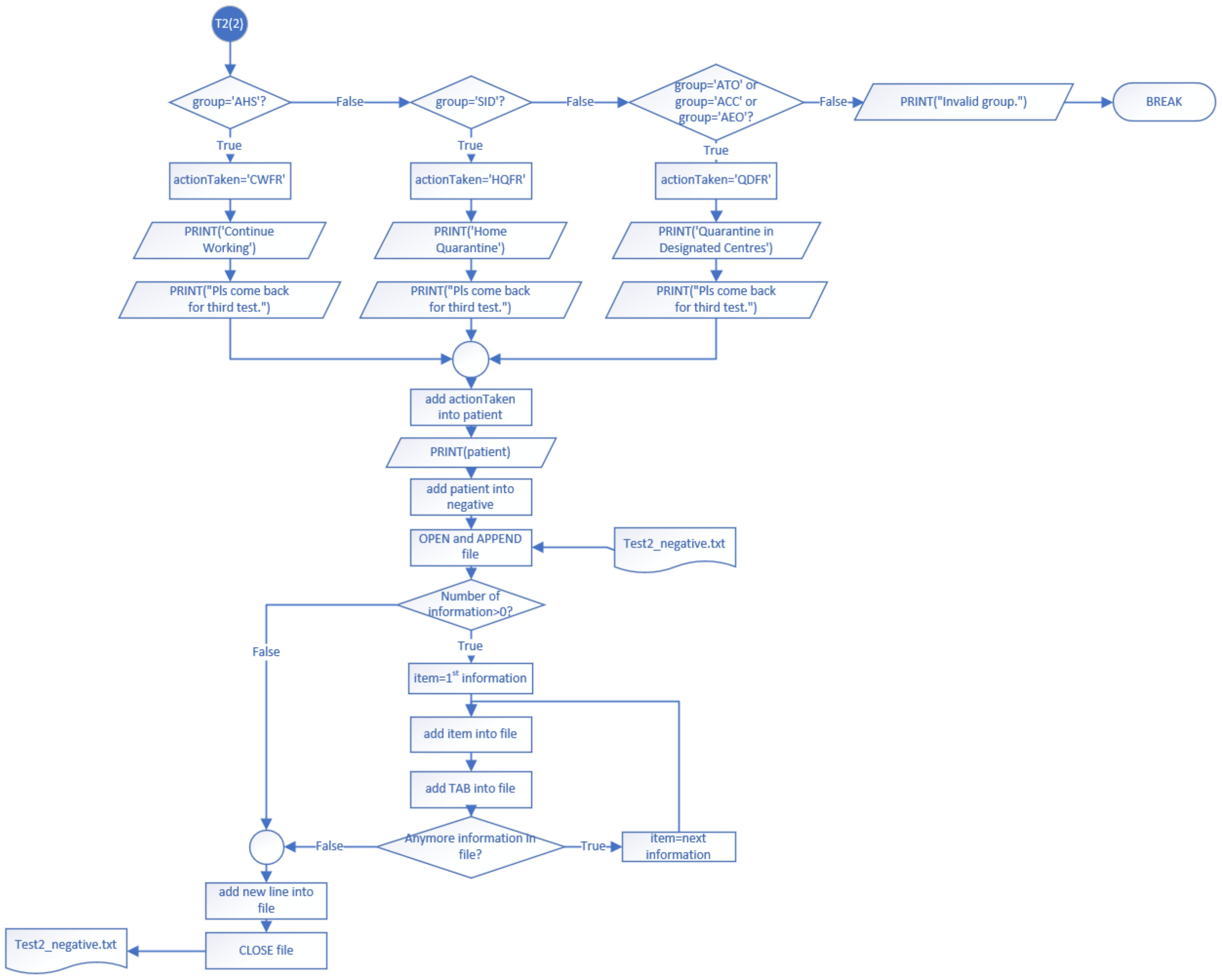


Figure 2.2.13 Test2 (Part2)

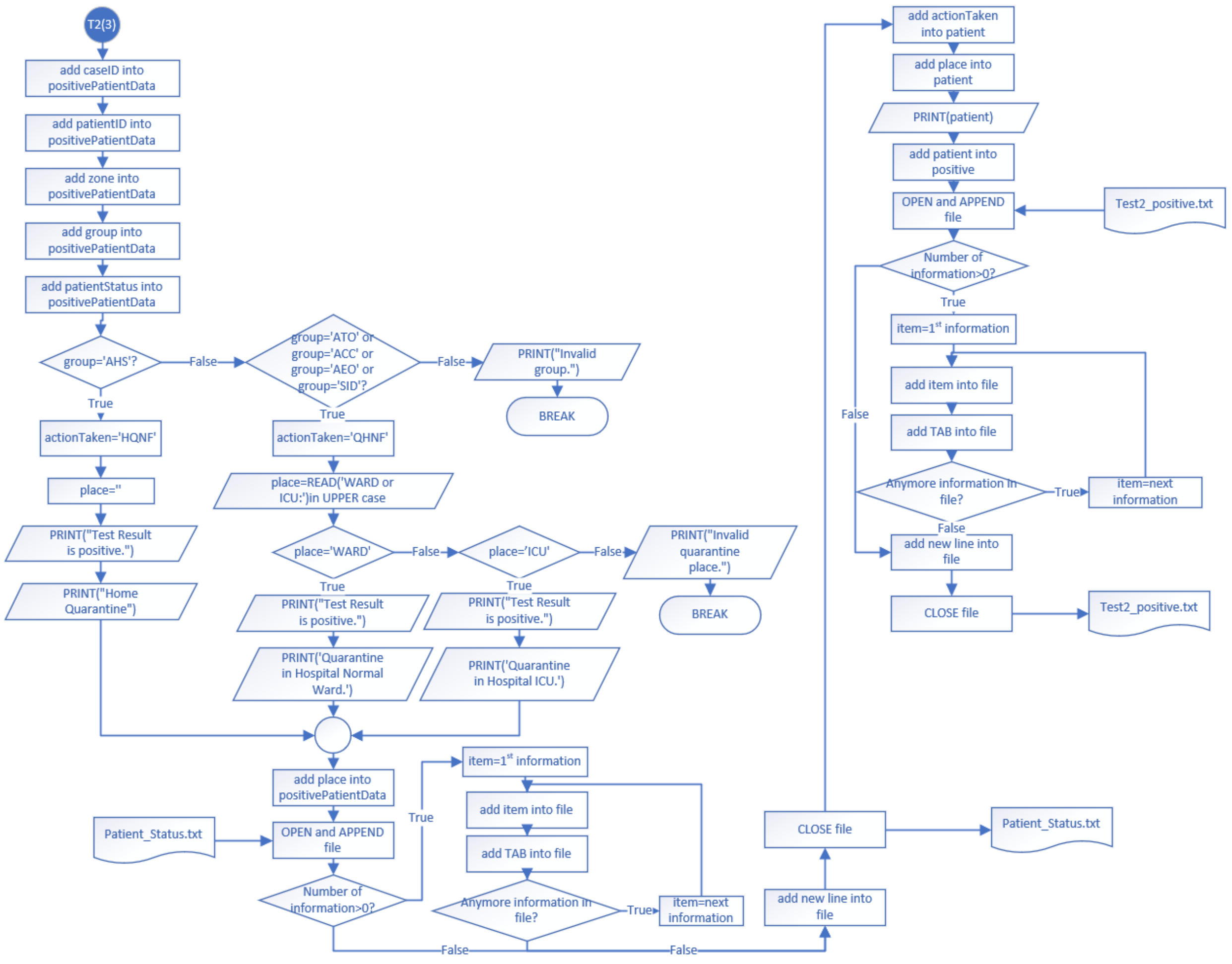

T2(3)
add caseID into positivePatientData
add patientID into positivePatientData
add zone into positivePatientData
add group into positivePatientData
add patientStatus into positivePatientData
group='AHS'?
True
False
actionTaken='HQNF'
place=''
PRINT("Test Result is positive.")
PRINT("Home Quarantine")
group='ATO' or group='ACC' or group='AEO' or group='SID'?
False
PRINT("Invalid group.")
BREAK
True
actionTaken='QHNF'
place=READ('WARD or ICU:')in UPPER case
place='WARD'
False
place='ICU'
False
PRINT("Invalid quarantine place.")
BREAK
True
PRINT("Test Result is positive.")
PRINT('Quarantine in Hospital Normal Ward.')
True
PRINT("Test Result is positive.")
PRINT('Quarantine in Hospital ICU.')
add place into positivePatientData
Patient_Status.txt
OPEN and APPEND file
Number of information>0?
True
False
item=1st information
add item into file
add TAB into file
Anymore information in file?
True
item=next information
False
add new line into file
CLOSE file
Patient_Status.txt
add actionTaken into patient
add place into patient
PRINT(patient)
add patient into positive
OPEN and APPEND file
Test2_positive.txt
Number of information>0?
True
False
item=1st information
add item into file
add TAB into file
Anymore information in file?
True
item=next information
False
add new line into file
CLOSE file
Test2_positive.txt

Figure 2.2.14 Test2 (Part3)

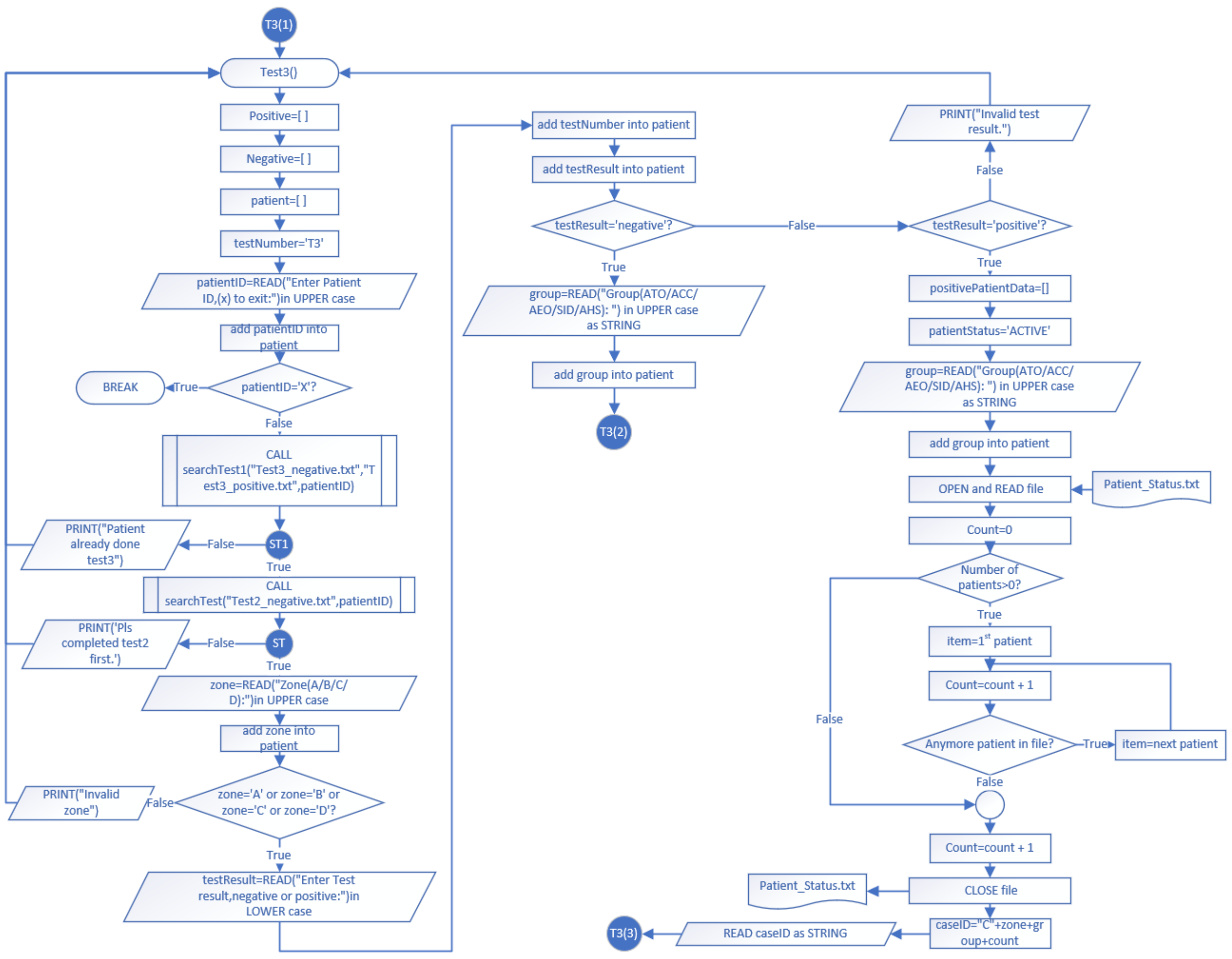

T3(1)
Test3()
Positive=[ ]
Negative=[ ]
patient=[ ]
testNumber='T3'
patientID=READ("Enter Patient ID,(x) to exit:")in UPPER case
add patientID into patient
patientID='X'?
True
BREAK
False
CALL searchTest1("Test3_negative.txt","Test3_positive.txt",patientID)
ST1
False
PRINT("Patient already done test3")
True
CALL searchTest("Test2_negative.txt",patientID)
ST
False
PRINT('Pls completed test2 first.')
True
zone=READ("Zone(A/B/C/D):")in UPPER case
add zone into patient
zone='A' or zone='B' or zone='C' or zone='D'?
False
PRINT("Invalid zone")
True
testResult=READ("Enter Test result,negative or positive:")in LOWER case
add testNumber into patient
add testResult into patient
testResult='negative'?
True
group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
add group into patient
T3(2)
False
testResult='positive'?
False
PRINT("Invalid test result.")
True
positivePatientData=[]
patientStatus='ACTIVE'
group=READ("Group(ATO/ACC/AEO/SID/AHS): ") in UPPER case as STRING
add group into patient
OPEN and READ file
Patient_Status.txt
Count=0
Number of patients>0?
True
item=1st patient
Count=count + 1
Anymore patient in file?
True
item=next patient
False
False
Count=count + 1
CLOSE file
Patient_Status.txt
caseID="C"+zone+group+count
READ caseID as STRING
T3(3)

Figure 2.2.15 Test3 (Part1)

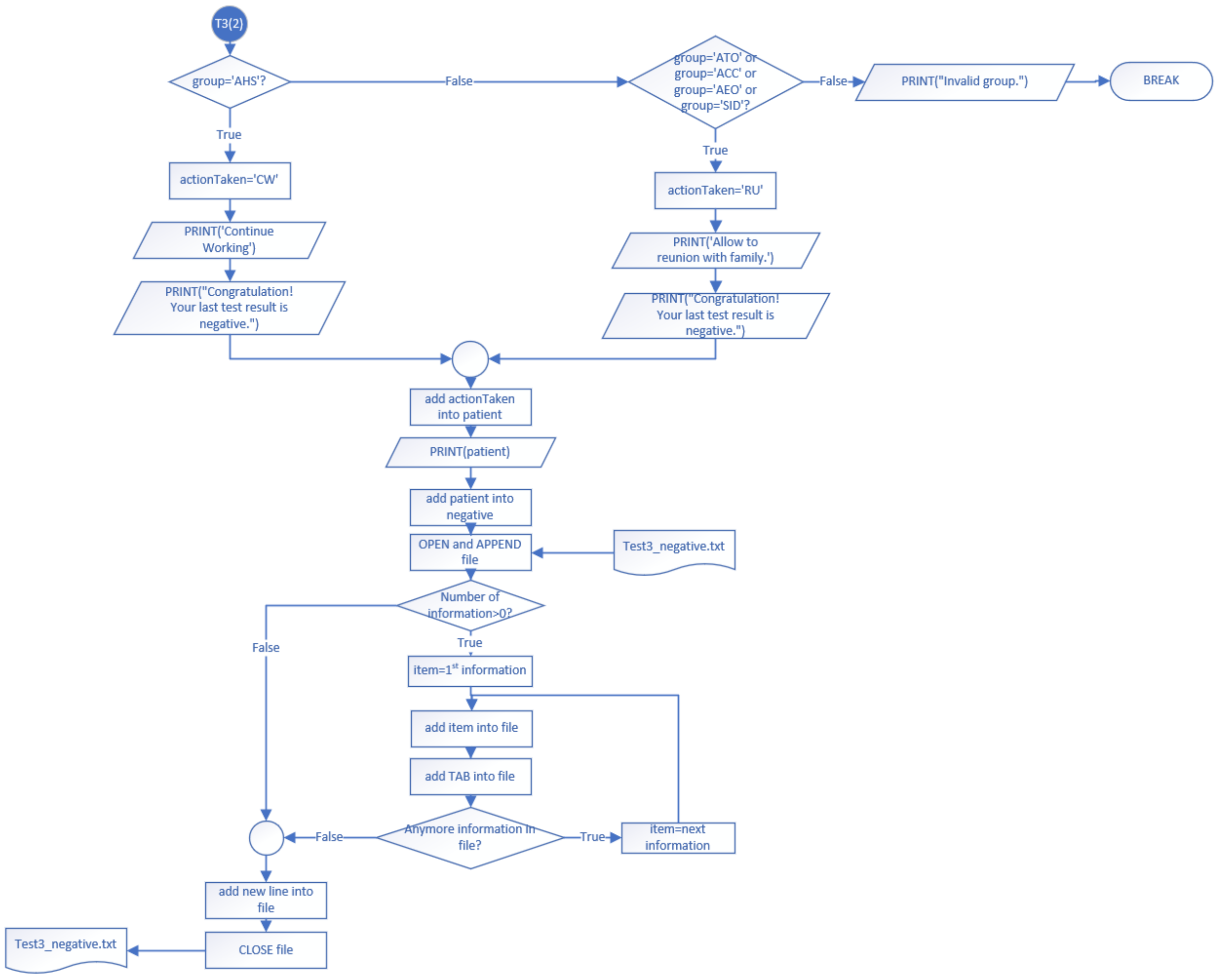

T3(2)
group='AHS'?
False
True
actionTaken='CW'
PRINT('Continue Working')
PRINT("Congratulation! Your last test result is negative.")
group='ATO' or group='ACC' or group='AEO' or group='SID'?
False
PRINT("Invalid group.")
BREAK
True
actionTaken='RU'
PRINT('Allow to reunion with family.')
PRINT("Congratulation! Your last test result is negative.")
add actionTaken into patient
PRINT(patient)
add patient into negative
OPEN and APPEND file
Test3_negative.txt
Number of information>0?
False
True
item=1st information
add item into file
add TAB into file
Anymore information in file?
False
True
item=next information
add new line into file
CLOSE file
Test3_negative.txt

Figure 2.2.16 Test3 (Part2)

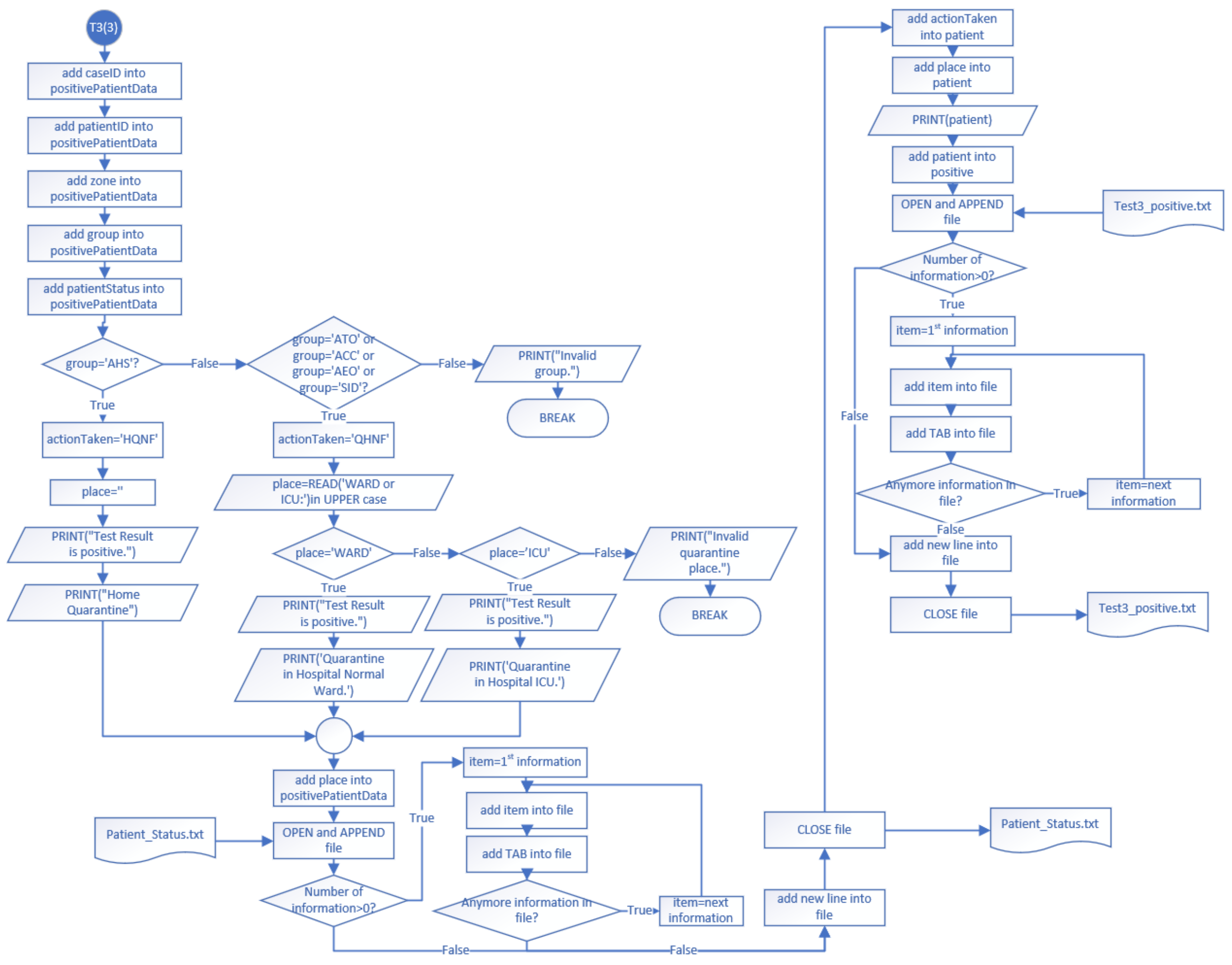

T3(3)
add caseID into positivePatientData
add patientID into positivePatientData
add zone into positivePatientData
add group into positivePatientData
add patientStatus into positivePatientData
group='AHS'?
False
True
actionTaken='HQNF'
place=''
PRINT("Test Result is positive.")
PRINT("Home Quarantine")
group='ATO' or group='ACC' or group='AEO' or group='SID'?
False
PRINT("Invalid group.")
BREAK
True
actionTaken='QHNF'
place=READ('WARD or ICU:')in UPPER case
place='WARD'
False
place='ICU'
False
PRINT("Invalid quarantine place.")
BREAK
True
PRINT("Test Result is positive.")
PRINT('Quarantine in Hospital Normal Ward.')
True
PRINT("Test Result is positive.")
PRINT('Quarantine in Hospital ICU.')
add place into positivePatientData
Patient_Status.txt
OPEN and APPEND file
Number of information>0?
True
item=1st information
add item into file
add TAB into file
Anymore information in file?
True
item=next information
False
False
add new line into file
CLOSE file
Patient_Status.txt
add actionTaken into patient
add place into patient
PRINT(patient)
add patient into positive
OPEN and APPEND file
Test3_positive.txt
Number of information>0?
True
False
item=1st information
add item into file
add TAB into file
Anymore information in file?
True
item=next information
False
add new line into file
CLOSE file
Test3_positive.txt

Figure 2.2.17 Test3 (Part3)

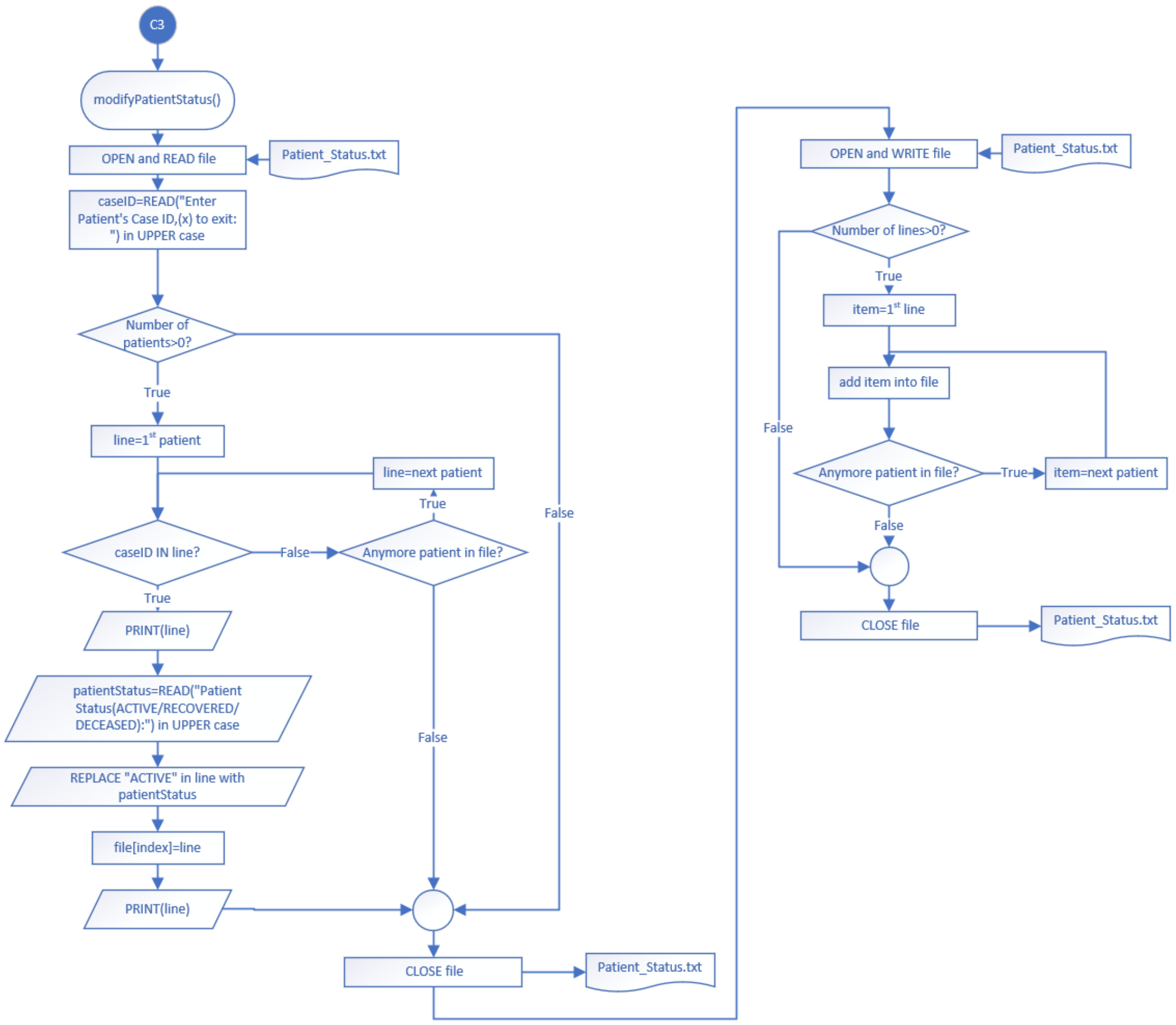


Figure 2.2.18 Modify Patient Status

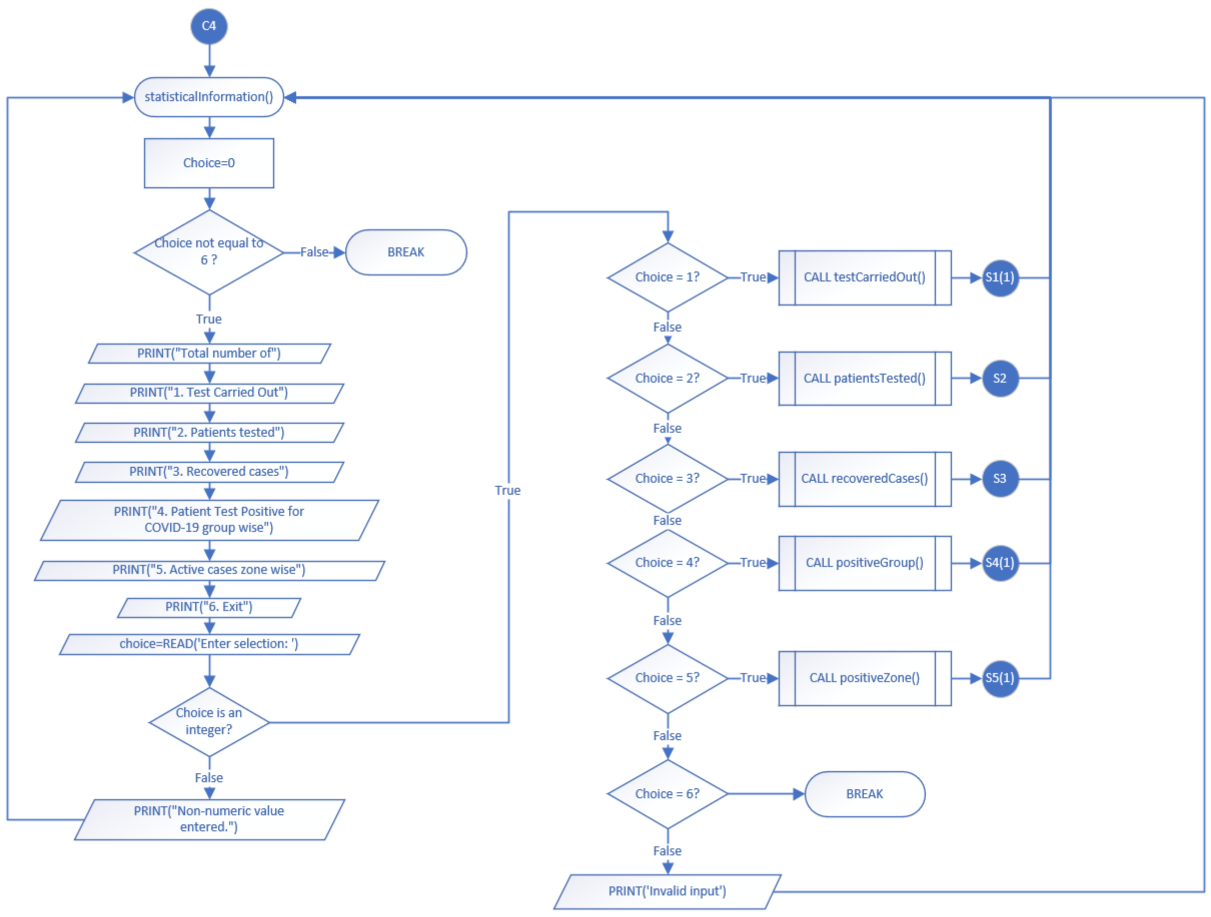


Figure 2.2.19 Statistical Information

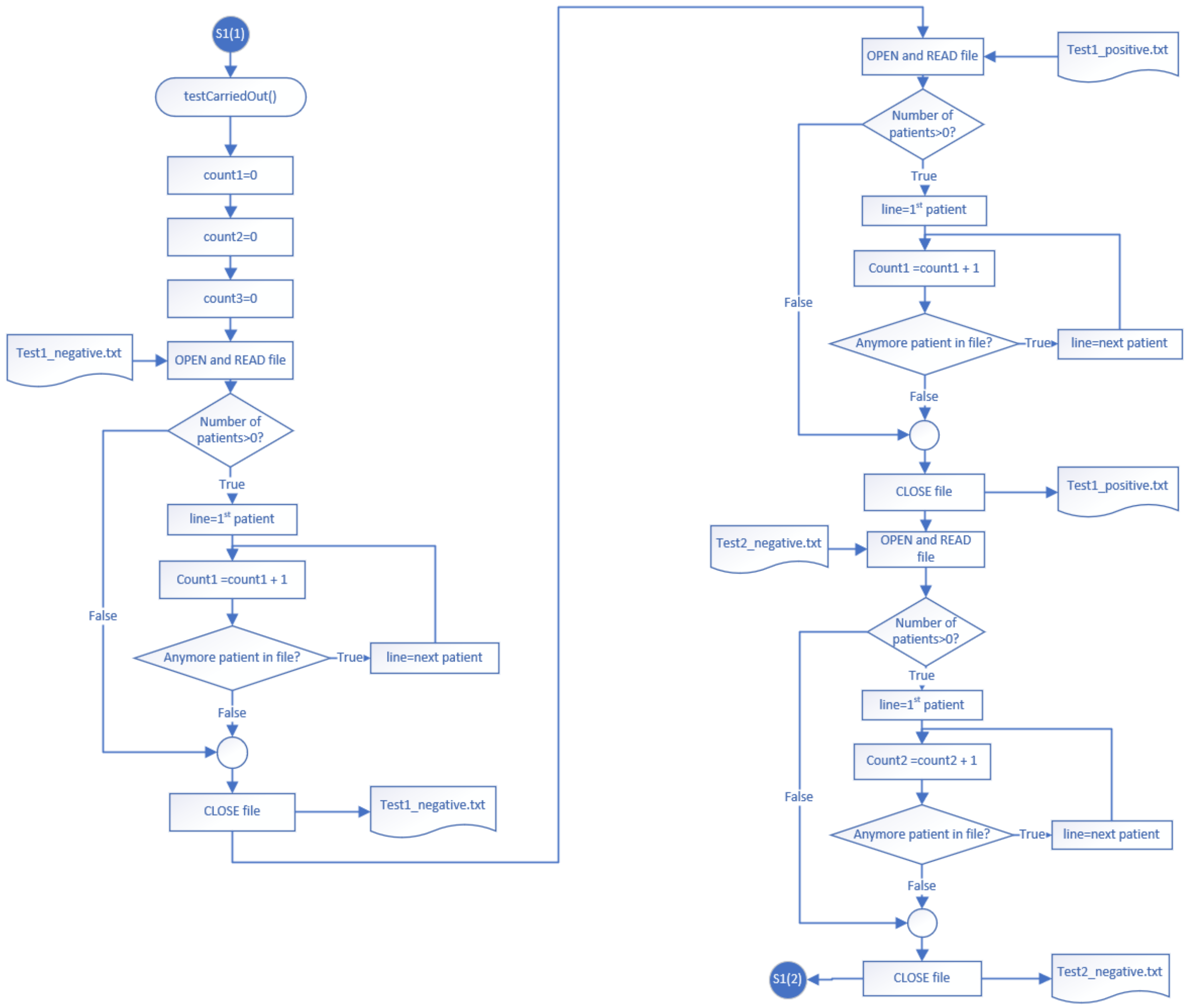


Figure 2.2.20 Test carried out (Part 1)

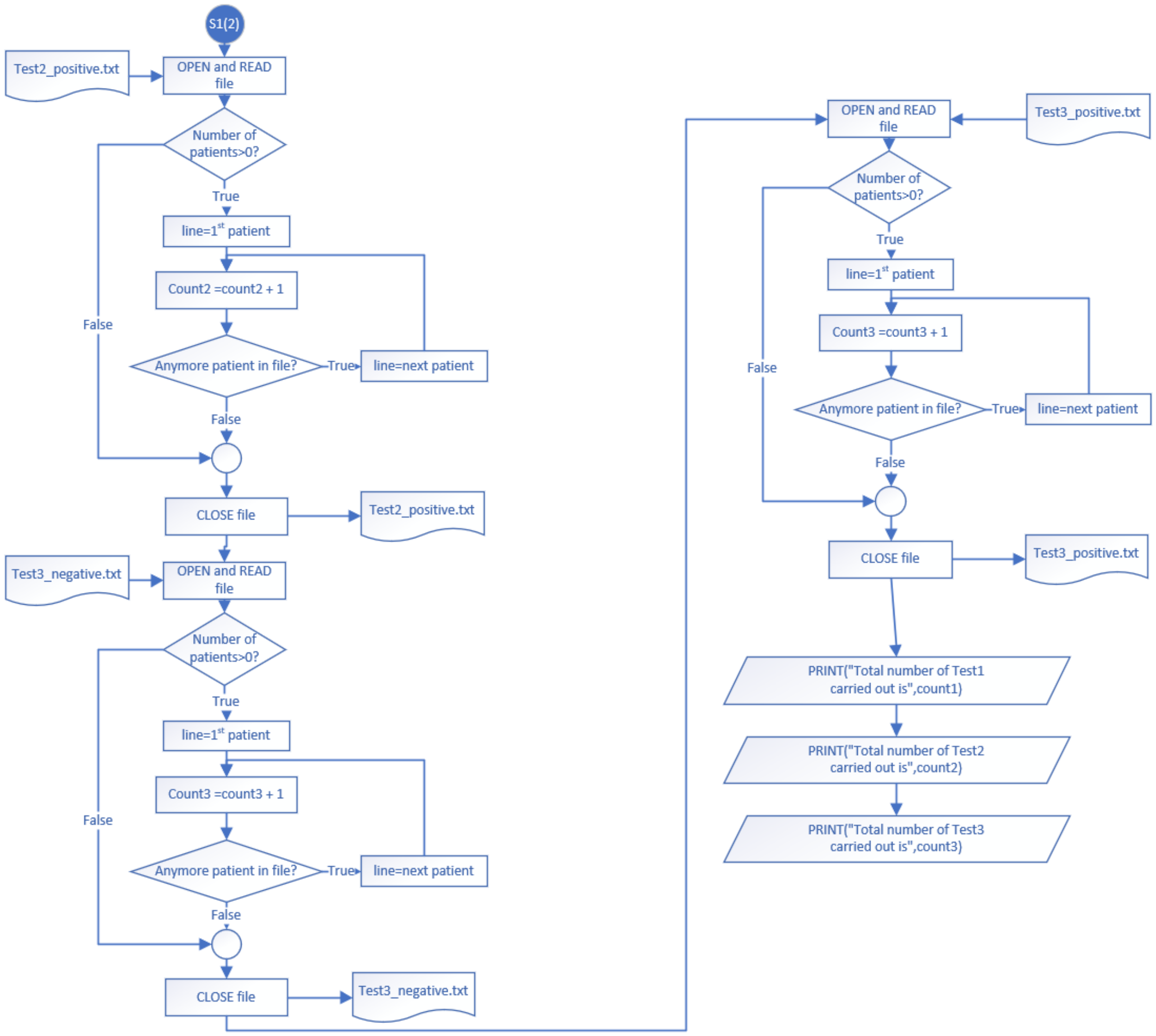


Figure 2.2.21 Test carried out (Part 2)

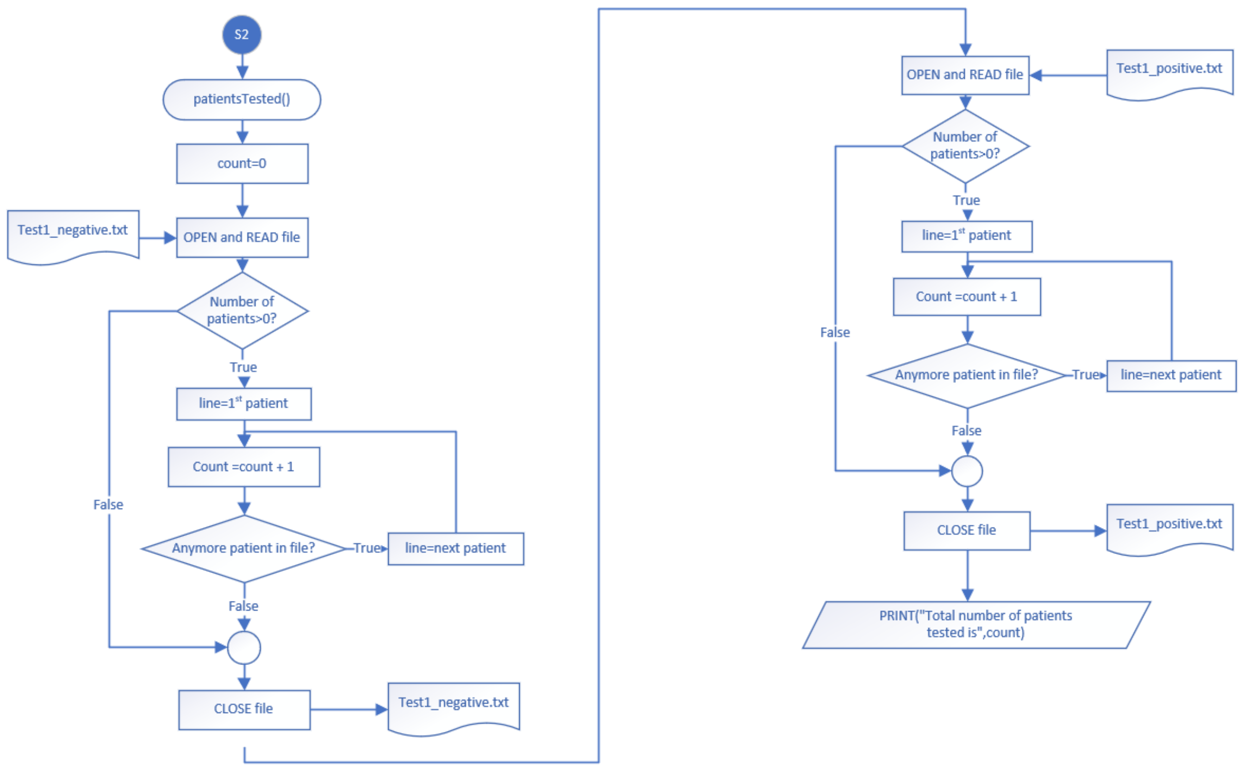


Figure 2.2.22 Patients tested

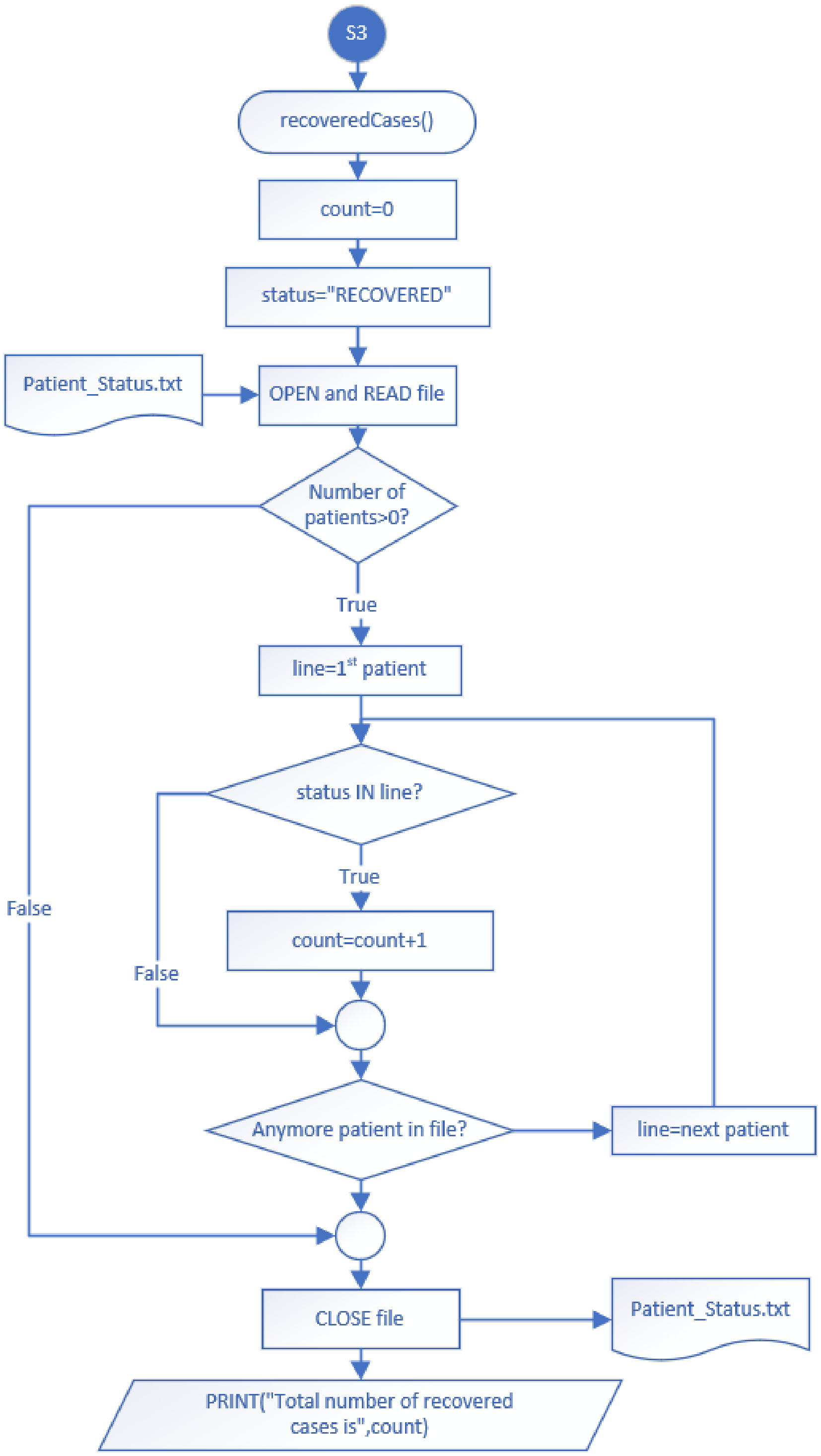


Figure 2.2.23 Recovered cases

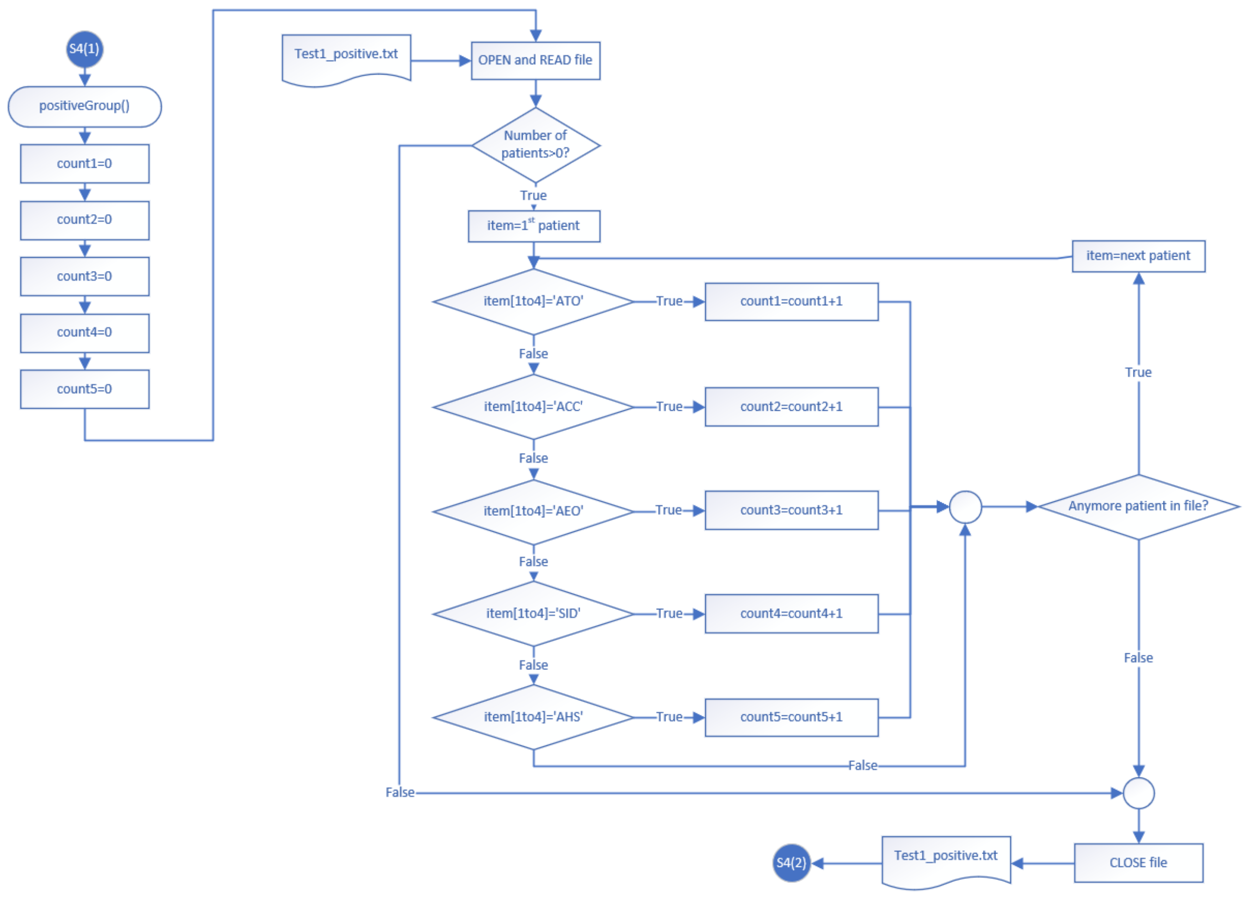


Figure 2.2.24 Patients test positive for COVID-19 group wise (Part 1)

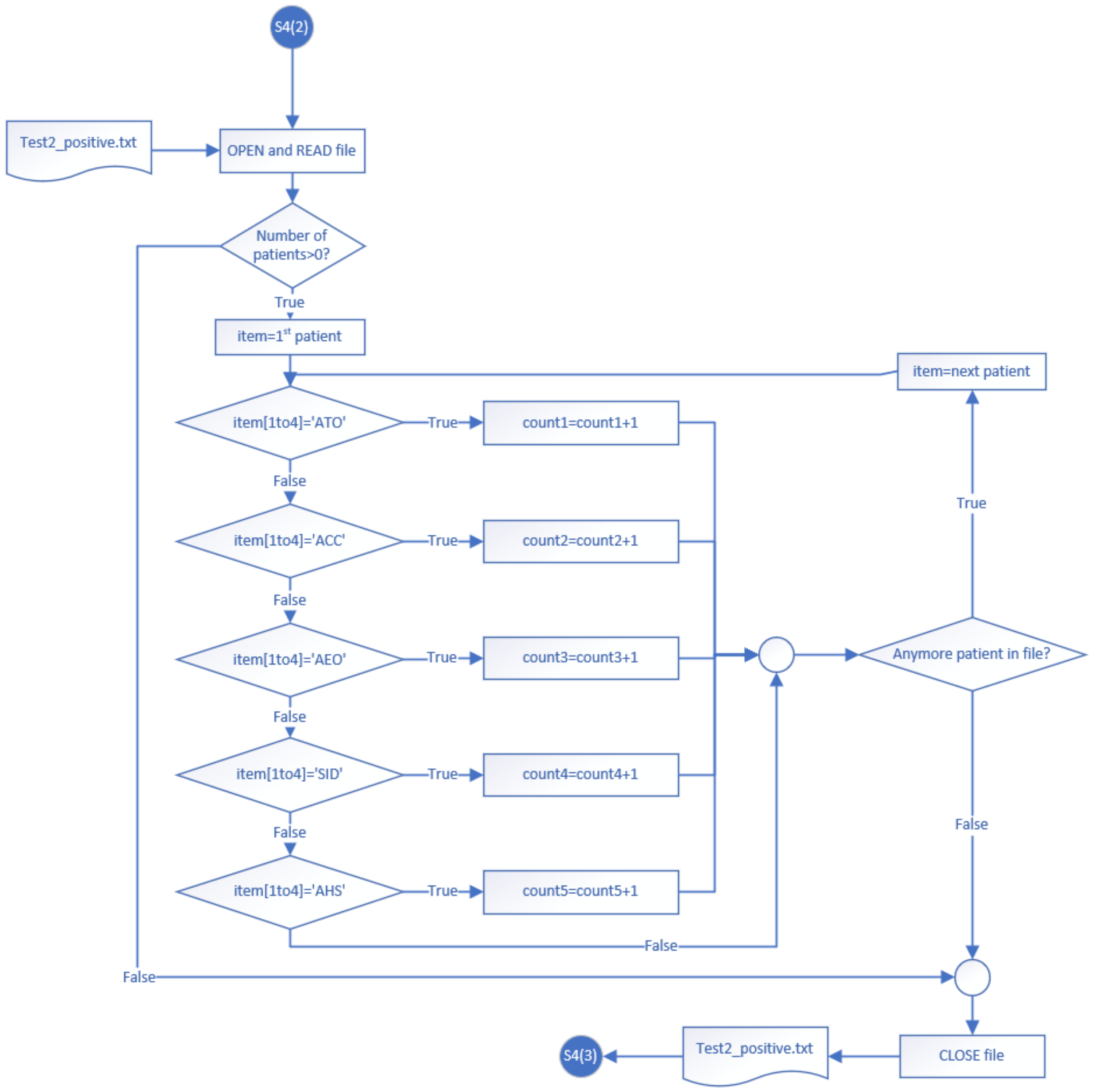


Figure 2.2.25 Patients test positive for COVID-19 group wise (Part 2)

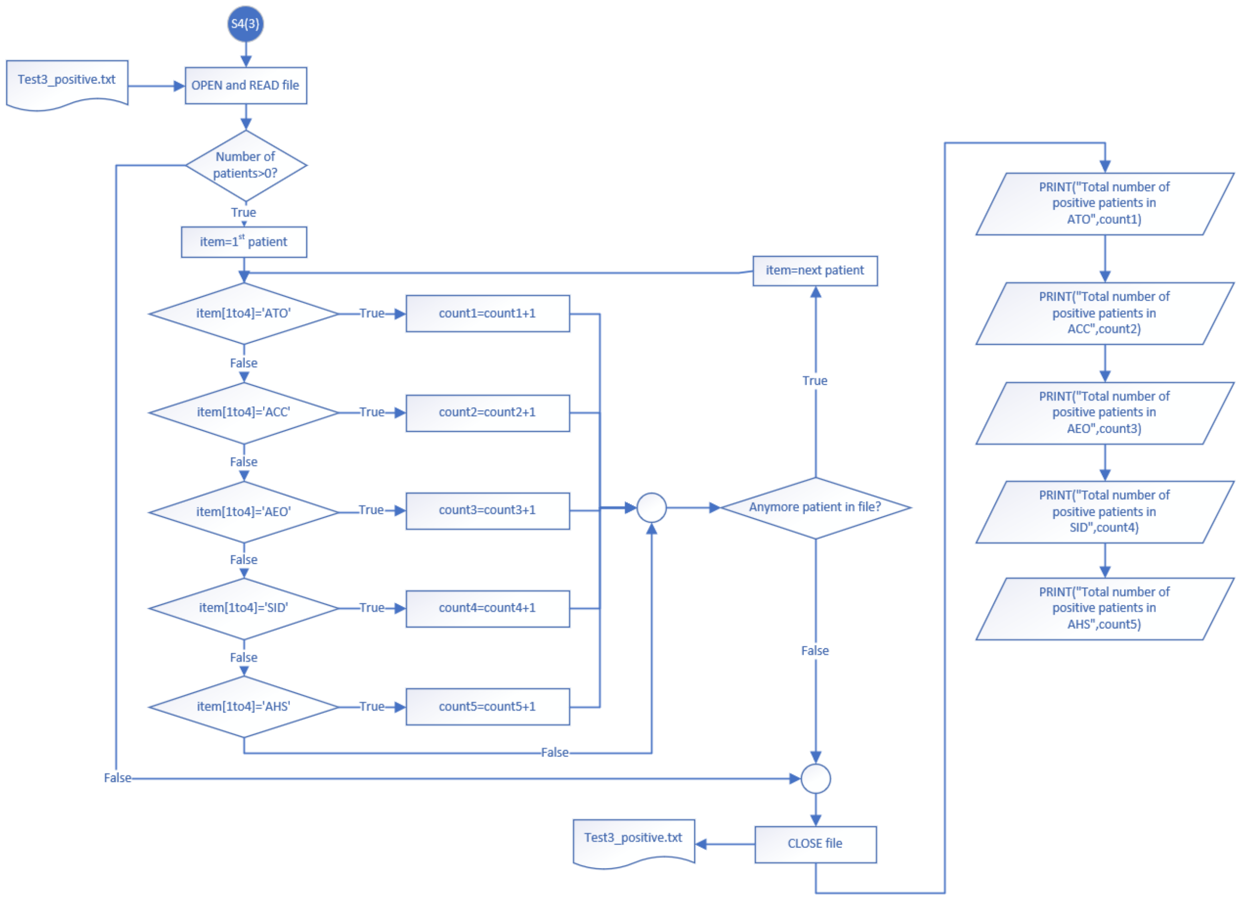


Figure 2.2.26 Patients test positive for COVID-19 group wise (Part 3)

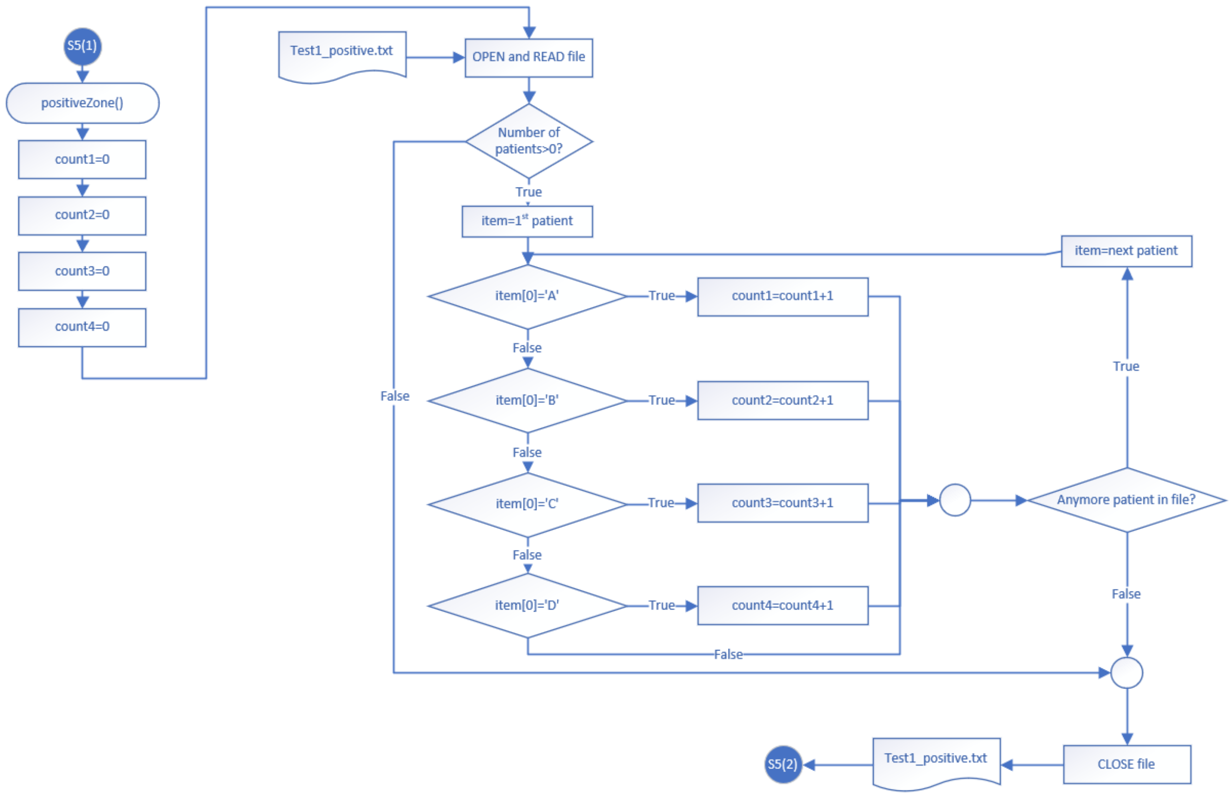


Figure 2.2.27 Active cases zone wise (Part 1)

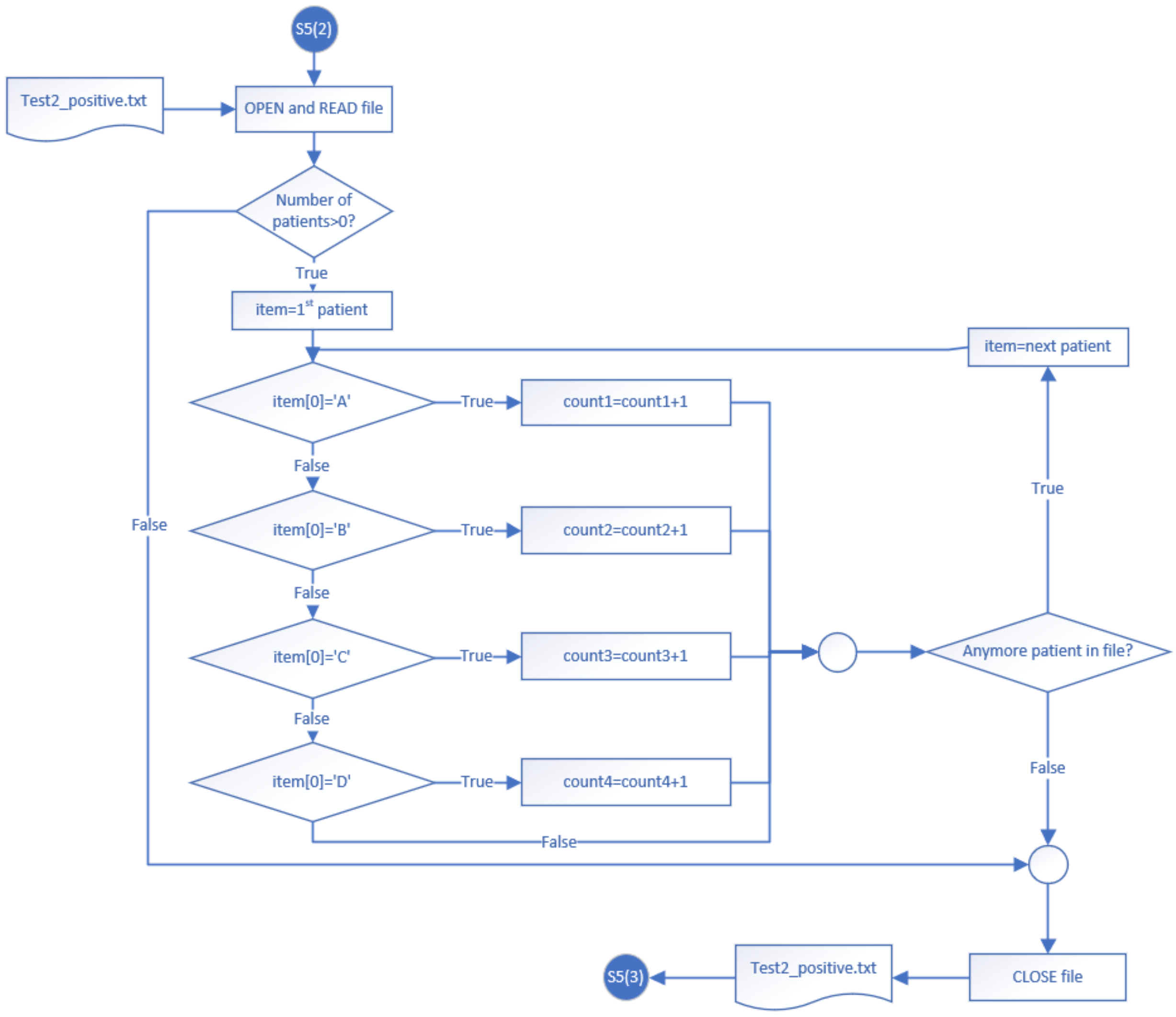


Figure 2.2.28 Active cases zone wise (Part 2)

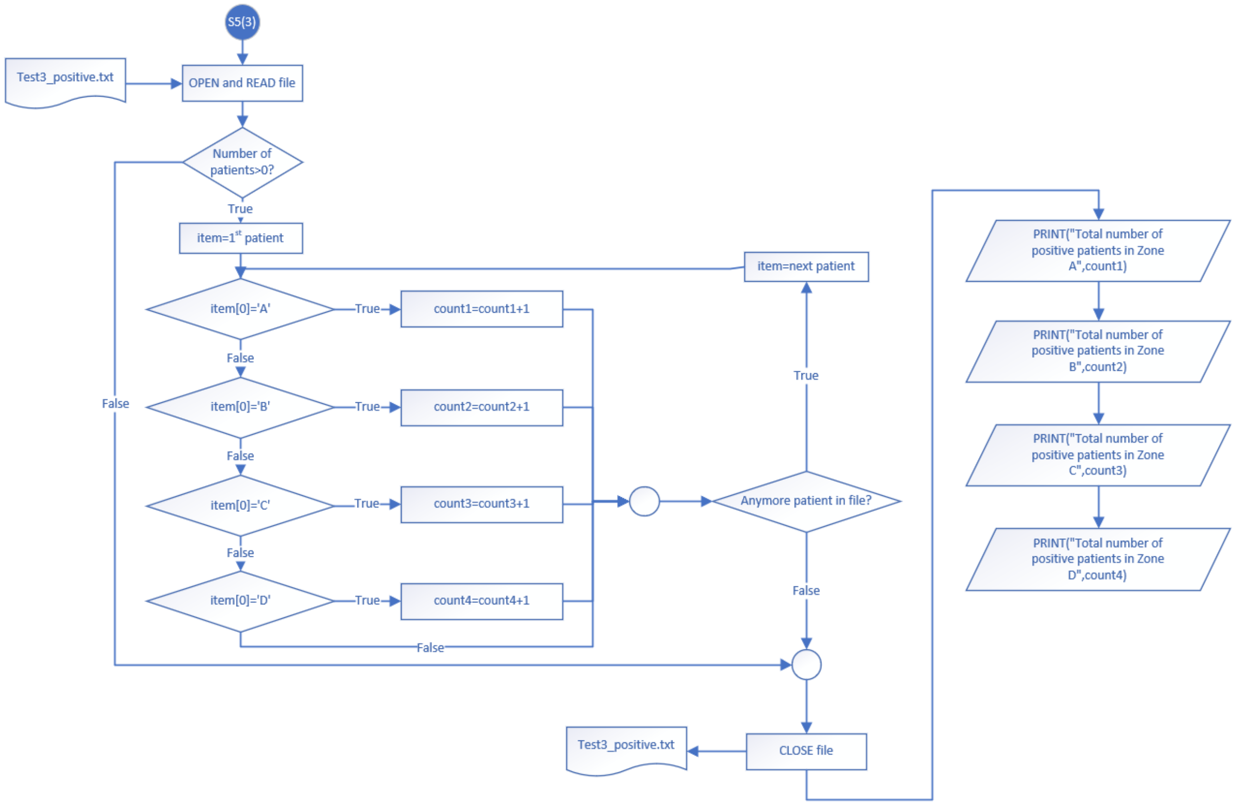


Figure 2.2.29 Active cases zone wise (Part 3)

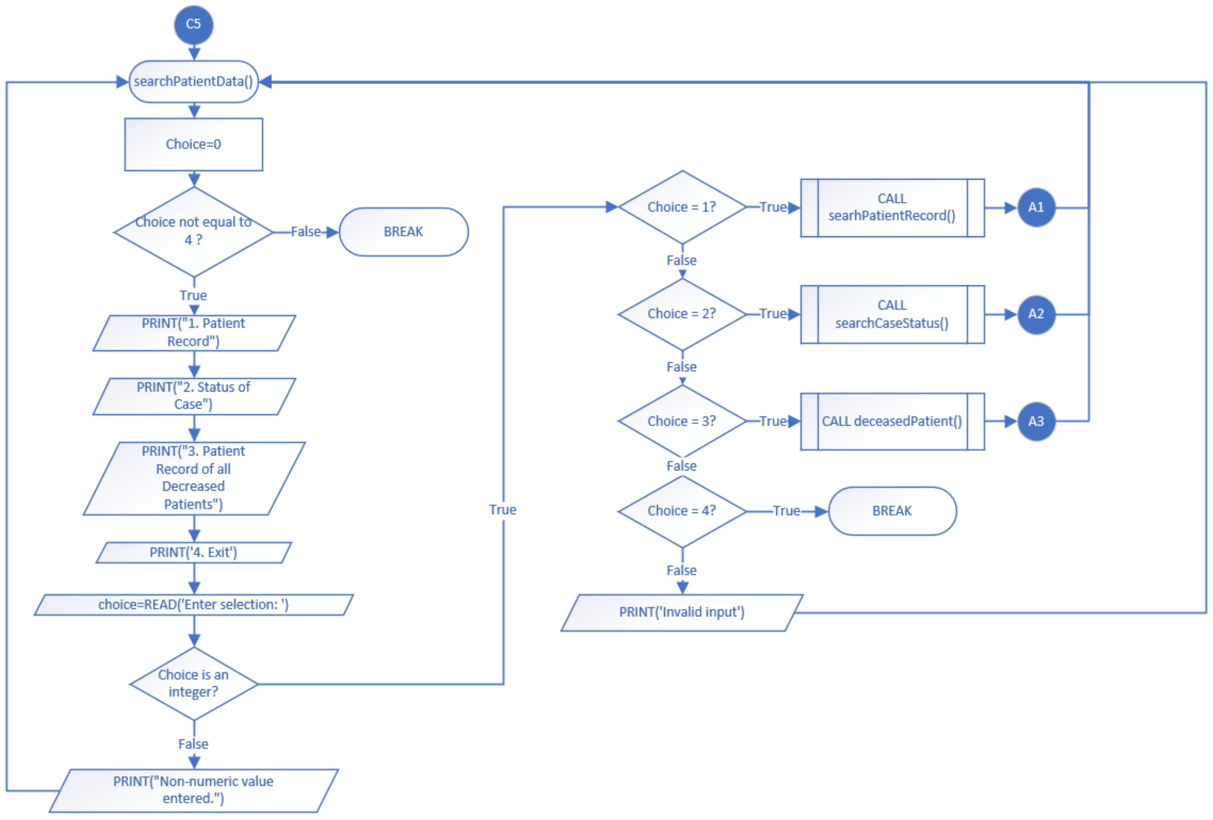


Figure 2.2.30 Searching Functionalities

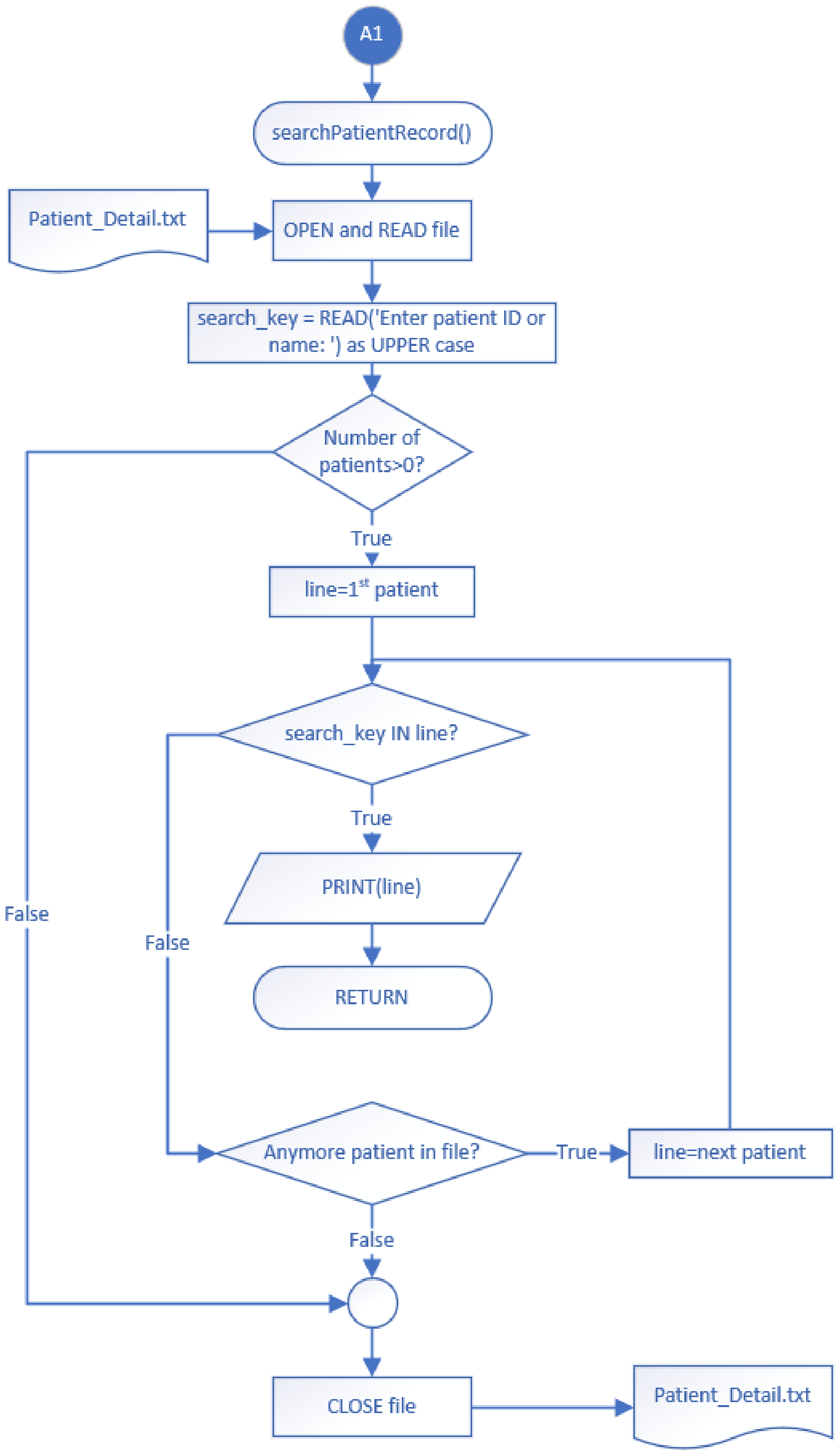


Figure 2.2.31 Search Patient Record

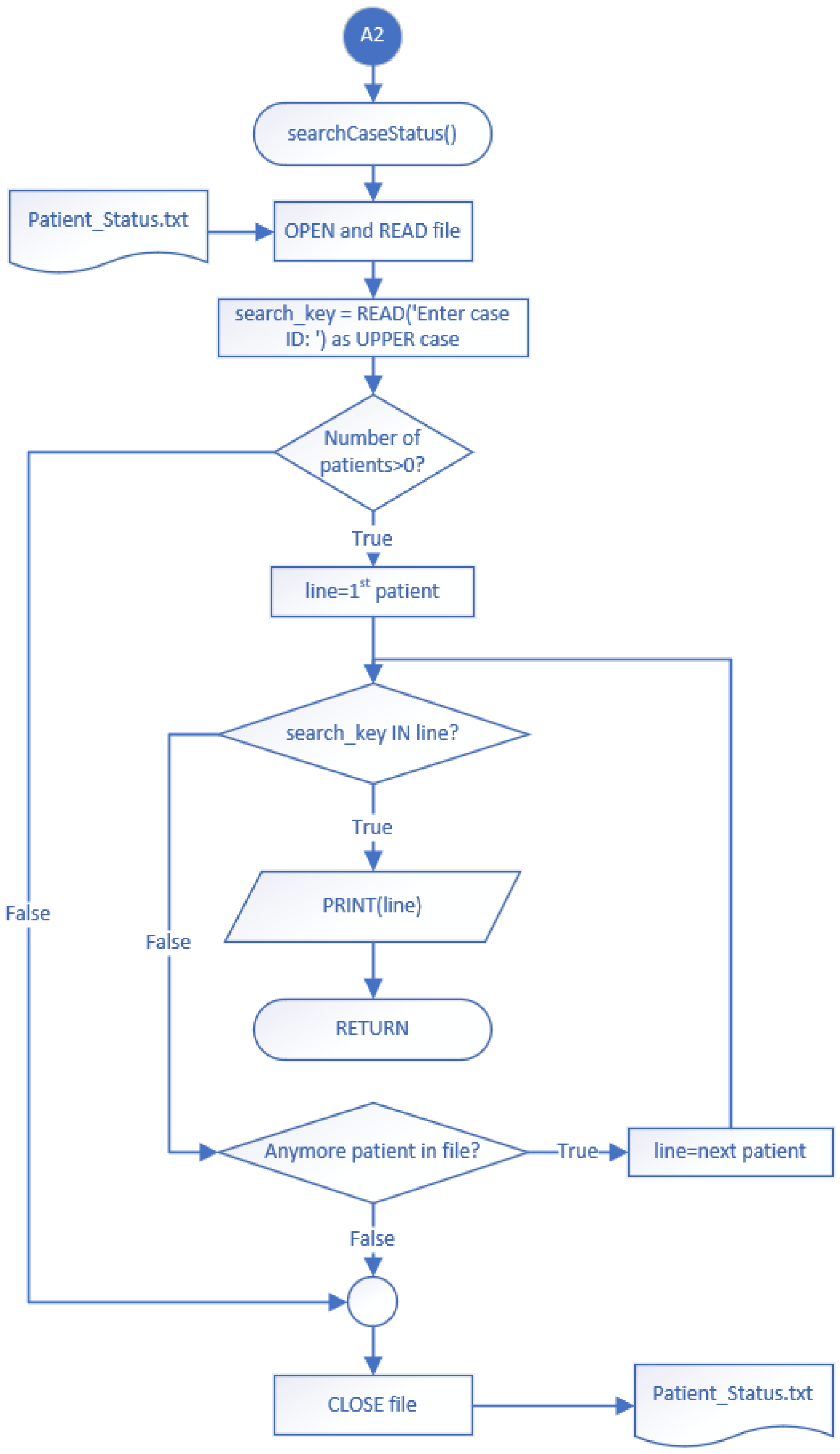


Figure 2.2.32 Search Case Detail

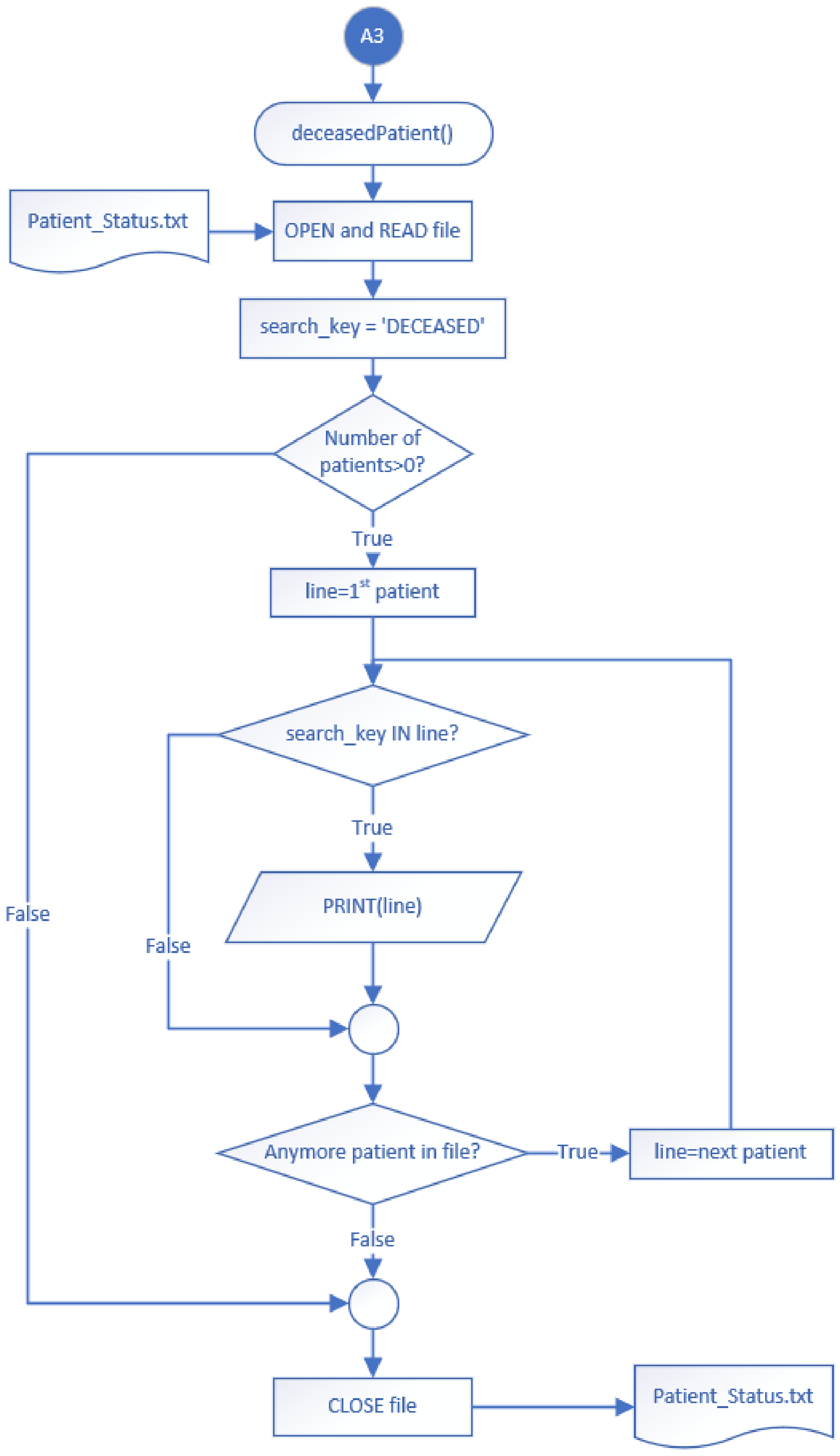


Figure 2.2.33 Search Deceased Patient Record

3.0 Program Source Code

```
#Check patients have done registration before or not
def searchPatient(fileName,patientName):
    fileHandler=open(fileName,"r")
    #This is a for loop to run through each line in fileHandler
    for line in fileHandler:
        if patientName in line:
            fileHandler.close()
            #If patient's name was found in the line, it will return False means that patient has done registration before
            return False
    fileHandler.close()
    #If patient's name was not found in the line, it will return True means that patient has not done registration before
    return True

#Patient registration and record in a text file
def patientRegistration():
    while True:
        patient=[]
        patientName=str(input("Patient Name,(x) to exit: ")).upper()
        if patientName == 'X':
            break
        if searchPatient("Patient_Detail.txt",patientName):
            #Collect patients information if patients didn't do registration before
            getID=open('Patient_Detail.txt','r+')
            count=0
            #This is a for loop to run through each line in getID and each line represents 1 patient
            for line in getID:
                count+=1
            #A unique number for new patient
            count=count+1
            getID.close()
            age=input("Age: ")
            try:
                #Age must be an integer
                age=str(int(age))
            except:
```

```
        print("Invalid age.")
        break
    print("Select a group:")
    print("ATO=Asymptomatic individuals with history of travelling overseas")
    print("ACC=Asymptomatic individuals with history of contact with known case of COVID-19")
    print("AEO=Asymptomatic individuals who had attended event associated with known COVID-19
outbreak")
    print("SID=Symptomatic individuals")
    print("AHS=Asymptomatic hospital staff")
    group=str(input("Group: ")).upper()
    #Group must be ATO, ACC, AEO, SID or AHS
    if group=='ATO' or group=='ACC' or group=='AEO' or group=='SID' or group=='AHS':
        pass
    else:
        print("Invalid group.")
        break
    print("Select a zone:")
    print("A-East")
    print("B-West")
    print("C-North")
    print("D-South")
    zone=str(input("Zone(A/B/C/D): ")).upper()
    #Zone must be A, B, C or D
    if zone=='A' or zone=='B' or zone=='C' or zone=='D':
        pass
    else:
        print("Invalid zone.")
        break
    contactNumber=input("Contact Number: ")
    #Contact number must be an integer
    try:
        contactNumber=str('0')+str(int(contactNumber))
    except:
        print("Invalid contact number.")
        break
    emailAddress=str(input("Email Address: "))
    #Join zone, group and count to form a unique patient ID
```

```
            patientID=str(zone)+str(group)+str(count)
            patient.append(patientID)
            patient.append(patientName)
            patient.append(age)
            patient.append(group)
            patient.append(zone)
            patient.append(contactNumber)
            patient.append(emailAddress)
            print(patient)
            #Append one more patient to the txt file
            fileHandler=open("Patient_Detail.txt","a")
            #This is a for loop to run through each items in patient and write them into fileHandler
            for items in patient:
                fileHandler.write(items)
                fileHandler.write('\t')
            fileHandler.write('\n')
            fileHandler.close()
        else:
            print("Patient Registered.")
        print()
    return


#-----------------------------------------------------------------------------------------------------------------------------------------------

#Check patients have done registration,test1 or test2 before or not
def searchTest(fileName,patientID):
    fileHandler=open(fileName,"r")
    #This is a for loop to run through each line in fileHandler
    for line in fileHandler:
        if patientID in line:
            print(line)
            fileHandler.close()
            return True
            #If patient's ID was found in the line, it will return True
            #In test1,patients must complete registration first
            #In test2,patients must complete test1 and the result is negative
            #In test3,patients must complete test2 and the result is negative
    fileHandler.close()
    #If patient's ID was not found in the line, it will return False
    return False

#Check patients have done test1, test2 or test3 before or not
def searchTest1(fileName1,fileName2,patientID):
    fileHandler=open(fileName1,"r")
    #This is a for loop to run through each line in fileHandler
    for line in fileHandler:
        if patientID in line:
            fileHandler.close()
            return False
            #If patient's ID was found in the line, it will return False
            #In test1,patients who already done test1 cannot run test1 again
            #In test2,patients who already done test2 cannot run test2 again
            #In test3,patients who already done test3 cannot run test3 again
    fileHandler.close()
    fileHandler=open(fileName2,"r")
    for line in fileHandler:
        if patientID in line:
            fileHandler.close()
            return False
    fileHandler.close()
    return True
    #If patient's ID was not found in the line, it will return True

#Choice for user to run test1, test2 or test3
def testResults():
    choice=0
    while (choice!=4):
        print('Select a test')
        print('1. Test 1')
        print('2. Test 2')
        print('3. Test 3')
        print('4. Exit')
        choice=input('Enter selection: ')
        #'choice' must be a number(1,2,3,4)
        try:
            choice=int(choice)
            if choice==1:
                test1()
            elif choice==2:
                test2()
            elif choice==3:
                test3()
            elif choice==4:
                break
            #If 'choice' is not a number between 1-4, it will go to 'else'
            else:
                print('Invalid input')
        #If 'choice' is not a number, it will go to 'except'
        except:
            print('Non-numeric value entered.')
        print()

#Test1
def test1():
    #'positive' and 'negative' will reset when run the function
    positive=[]
    negative=[]
    #The program will run/continue only the statement is 'True'
    while True:
        patient=[]
        testNumber='T1'
        patientID=input("Enter Patient ID,(x) to exit:").upper()
        patient.append(patientID)
        if patientID=='X':
            break
        #To check whether patients done test1 before or not
        if searchTest1("Test1_negative.txt","Test1_positive.txt",patientID):
            #To check whether patients done registration before or not
            if searchTest("Patient_Detail.txt",patientID):
                zone=input("Zone(A/B/C/D):").upper()
```

```
patient.append(zone)
#Zone must be A, B, C or D
if zone=='A' or zone=='B' or zone=='C' or zone=='D':
    testResult=input("Enter Test result,negative or positive:").lower()
    patient.append(testNumber)
    patient.append(testResult)
    if testResult=='negative':
        group=str(input("Group(ATO/ACC/AEO/SID/AHS): ")).upper()
        patient.append(group)
        #Group must be ATO, ACC, AEO, SID or AHS
        if group=='AHS':
            actionTaken='CWFR'
            print('Continue Working')
            print("Pls come back for second test.")
        elif group=='SID':
            actionTaken='HQFR'
            print('Home Quarantine')
            print("Pls come back for second test.")
        elif group=='ATO' or group=='ACC' or group=='AEO':
            actionTaken='QDFR'
            print('Quarantine in Designated Centres')
            print("Pls come back for second test.")
        else:
            print("Invalid group.")
            break
        patient.append(actionTaken)
        print(patient)
        negative.append(patient)
        fileHandler=open('Test1_negative.txt','a')
        #This is a for loop to run through each item in negative for writing item into fileHandler
        for items in negative:
            for item in items:
                fileHandler.write(item)
                fileHandler.write('\t')
            fileHandler.write('\n')
        fileHandler.close()
    elif testResult=='positive':
```

```
#'positivePatientData' will reset when run test result becomes positive
positivePatientData=[]
patientStatus='ACTIVE'
group=str(input("Group(ATO/ACC/AEO/SID/AHS): ")).upper()
patient.append(group)
getID=open('Patient_Status.txt','r+')
count=0
#This is a for loop to run through each line in getID and each line represents 1 patient
for line in getID:
   count+=1
#A unique number for new patient
count=count+1
getID.close()
#join 'C',zone,group and count to form a unique case ID
caseID=str("C")+str(zone)+str(group)+str(count)
positivePatientData.append(caseID)
positivePatientData.append(patientID)
positivePatientData.append(zone)
positivePatientData.append(group)
positivePatientData.append(patientStatus)
if group=='AHS':
   actionTaken='HQNF'
   place=""
   print("Test Result is positive.")
   print("Home Quarantine")
elif group=='ATO' or group=='ACC' or group=='AEO'or group=='SID':
   actionTaken='QHNF'
   place=input('WARD or ICU:').upper()
   #Place must be WARD or ICU
   if place=='WARD':
      print("Test Result is positive.")
      print('Quarantine in Hospital Normal Ward.')
   elif place=='ICU':
      print("Test Result is positive.")
      print('Quarantine in Hospital ICU.')
   else:
      print("Invalid quarantine place.")
```

```
                                break
                        else:
                            print("Invalid group.")
                            break
                        positivePatientData.append(place)
                        fileHandler=open('Patient_Status.txt','a')
                        for items in positivePatientData:
                            fileHandler.write(items)
                            fileHandler.write('\t')
                        fileHandler.write('\n')
                        fileHandler.close()
                        patient.append(actionTaken)
                        patient.append(place)
                        print(patient)
                        positive.append(patient)
                        fileHandler=open("Test1_positive.txt","a")
                        for items in positive:
                            for item in items:
                                fileHandler.write(item)
                                fileHandler.write('\t')
                            fileHandler.write('\n')
                        fileHandler.close()
                    else:
                        print("Invalid test result.")
                else:
                    print("Invalid zone")
            else:
                print("Patient not found.")
        else:
            print("Patient already done test1")
        print()
        break

#Test2
def test2():
    #'positive' and 'negative' will reset when run the function
    positive=[]
```

```
negative=[]
#The program will run/continue only the statement is 'True'
while True:
    patient=[]
    testNumber='T2'
    patientID=input("Enter Patient ID,(x) to exit:").upper()
    patient.append(patientID)
    if patientID=='X':
        break
    #To check whether patients done test2 before or not
    if searchTest1("Test2_positive.txt","Test2_negative.txt",patientID):
        #To check whether patients done test1 before or not and the result must be negative
        if searchTest("Test1_negative.txt",patientID):
            zone=input("Zone(A/B/C/D):").upper()
            patient.append(zone)
            #Zone must be A, B, C or D
            if zone=='A' or zone=='B' or zone=='C' or zone=='D':
                testResult=input("Enter Test result,negative or positive:").lower()
                patient.append(testNumber)
                patient.append(testResult)
                if testResult=='negative':
                    group=str(input("Group(ATO/ACC/AEO/SID/AHS): ")).upper()
                    patient.append(group)
                    #Group must be ATO, ACC, AEO, SID or AHS
                    if group=='AHS':
                        actionTaken='CWFR'
                        print('Continue Working')
                        print("Pls come back for third test.")
                    elif group=='SID':
                        actionTaken='HQFR'
                        print('Home Quarantine')
                        print("Pls come back for third test.")
                    elif group=='ATO' or group=='ACC' or group=='AEO':
                        actionTaken='QDFR'
                        print('Quarantine in Designated Centres')
                        print("Pls come back for third test.")
                    else:
```

```
        print("Invalid group.")
        break
    patient.append(actionTaken)
    print(patient)
    negative.append(patient)
    fileHandler=open('Test2_negative.txt','a')
    #This is a for loop to run through each item in negative for writing item into fileHandler
    for items in negative:
        for item in items:
            fileHandler.write(item)
            fileHandler.write('\t')
        fileHandler.write('\n')
    fileHandler.close()
elif testResult=='positive':
    #'positivePatientData' will reset when run test result becomes positive
    positivePatientData=[]
    patientStatus='ACTIVE'
    group=str(input("Group(ATO/ACC/AEO/SID/AHS): ")).upper()
    patient.append(group)
    getID=open('Patient_Status.txt','r+')
    count=0
    #This is a for loop to run through each line in getID and each line represents 1 patient
    for line in getID:
        count+=1
    #A unique number for new patient
    count=count+1
    getID.close()
    #join 'C',zone,group and count to form a unique case ID
    caseID=str("C")+str(zone)+str(group)+str(count)
    positivePatientData.append(caseID)
    positivePatientData.append(patientID)
    positivePatientData.append(zone)
    positivePatientData.append(group)
    positivePatientData.append(patientStatus)
    if group=='AHS':
        actionTaken='HQNF'
        place=""
        print("Test Result is positive.")
        print('Home Quarantine')
    elif group=='ATO' or group=='ACC' or group=='AEO'or group=='SID':
        actionTaken='QHNF'
        place=input('WARD or ICU:').upper()
        #Place must be WARD or ICU
        if place=='WARD':
            print("Test Result is positive.")
            print('Quarantine in Hospital Normal Ward.')
        elif place=='ICU':
            print("Test Result is positive.")
            print('Quarantine in Hospital ICU.')
        else:
            print("Invalid quarantine place.")
            break
    else:
        print("Invalid group.")
        break
    positivePatientData.append(place)
    fileHandler=open('Patient_Status.txt','a')
    for items in positivePatientData:
        fileHandler.write(items)
        fileHandler.write('\t')
    fileHandler.write('\n')
    fileHandler.close()
    patient.append(actionTaken)
    patient.append(place)
    print(patient)
    positive.append(patient)
    fileHandler=open('Test2_positive.txt','a')
    for items in positive:
        for item in items:
            fileHandler.write(item)
            fileHandler.write('\t')
        fileHandler.write('\n')
    fileHandler.close()
else:
```

```
                        print("Invalid test result.")
                else:
                    print("Invalid zone")
            else:
                print('Pls completed registration or test1 first.')
                print('Positive patients did not need to run test2')
        else:
            print("Patient already done test2")
        print()
        break

#Test3
def test3():
    #'positive' and 'negative' will reset when run the function
    positive=[]
    negative=[]
    #The program will run/continue only the statement is 'True'
    while True:
        patient=[]
        testNumber='T3'
        patientID=input("Enter Patient ID,(x) to exit:").upper()
        patient.append(patientID)
        if patientID=='X':
            break
        #To check whether patients done test3 before or not
        if searchTest1("Test3_positive.txt","Test3_negative.txt",patientID):
            #To check whether patients done test2 before or not and the result must be negative
            if searchTest("Test2_negative.txt",patientID):
                zone=input("Zone(A/B/C/D):").upper()
                patient.append(zone)
                #Zone must be A, B, C or D
                if zone=='A' or zone=='B' or zone=='C' or zone=='D':
                    testResult=input("Enter Test result,negative or positive:").lower()
                    patient.append(testNumber)
                    patient.append(testResult)
                    if testResult=='negative':
                        group=str(input("Group(ATO/ACC/AEO/SID/AHS): ")).upper()
    patient.append(group)
    #Group must be ATO, ACC, AEO, SID or AHS
    if group=='AHS':
        actionTaken='CW'
        print('Continue Working')
        print("Congratulation!Your last test result is negative.")
    elif group=='ATO' or group=='ACC' or group=='AEO' or group=='SID':
        actionTaken='RU'
        print('Allow to reunion with family.')
        print("Congratulation!Your last test result is negative.")
    else:
        print("Invalid group.")
        break
    patient.append(actionTaken)
    print(patient)
    negative.append(patient)
    fileHandler=open('Test3_negative.txt','a')
    #This is a for loop to run through each item in negative for writing item into fileHandler
    for items in negative:
        for item in items:
            fileHandler.write(item)
            fileHandler.write('\t')
        fileHandler.write('\n')
    fileHandler.close()
elif testResult=='positive':
    #'positivePatientData' will reset when run test result becomes positive
    positivePatientData=[]
    patientStatus='ACTIVE'
    group=str(input("Group(ATO/ACC/AEO/SID/AHS): ")).upper()
    patient.append(group)
    getID=open('Patient_Status.txt','r+')
    count=0
    #This is a for loop to run through each line in getID and each line represents 1 patient
    for line in getID:
        count+=1
    #A unique number for new patient
    count=count+1
```

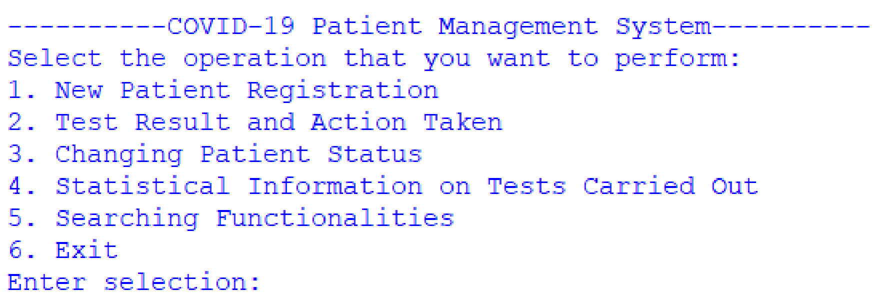
```
getID.close()
#join 'C',zone,group and count to form a unique case ID
caseID=str("C")+str(zone)+str(group)+str(count)
positivePatientData.append(caseID)
positivePatientData.append(patientID)
positivePatientData.append(zone)
positivePatientData.append(group)
positivePatientData.append(patientStatus)
if group=='AHS':
   actionTaken='HQNF'
   place=""
   print("Test Result is positive.")
   print('Home Quarantine')
elif group=='ATO' or group=='ACC' or group=='AEO'or group=='SID':
   actionTaken='QHNF'
   place=input('WARD or ICU:').upper()
   #Place must be WARD or ICU
   if place=='WARD':
      print("Test Result is positive.")
      print('Quarantine in Hospital Normal Ward.')
   elif place=='ICU':
      print("Test Result is positive.")
      print('Quarantine in Hospital ICU.')
   else:
      print("Invalid quarantine place.")
      break
else:
   print("Invalid group.")
   break
positivePatientData.append(place)
fileHandler=open('Patient_Status.txt','a')
for items in positivePatientData:
   fileHandler.write(items)
   fileHandler.write('\t')
fileHandler.write('\n')
fileHandler.close()
patient.append(actionTaken)
```

```
                patient.append(place)
                print(patient)
                positive.append(patient)
                fileHandler=open('Test3_positive.txt','a')
                for items in positive:
                    for item in items:
                        fileHandler.write(item)
                        fileHandler.write('\t')
                    fileHandler.write('\n')
                fileHandler.close()
            else:
                print("Invalid test result.")
        else:
            print("Invalid zone")
    else:
        print('Pls completed test2 first.')
        print('Positive patients did not need to run test3')
else:
    print("Patient already done test3")
print()
break

#---------------------------------------------------------------------------------------------------------------------------------------------

#Modify active patients' status
def modifyPatientStatus():
    fileHandler=open('Patient_Status.txt','r')
    caseID=input("Enter Patient's Case ID,(x) to exit: ").upper()
    #Read fileHandler and store information in fileData temporarily
    fileData=fileHandler.readlines()
    #Give a specific representation in each line in fileData
    for index,line in enumerate(fileData):
        line=line.strip()
        if caseID in line:
            #Print patient status if case ID was found in the line of fileData or it will automatically back to main menu
            print(line)
            patientStatus=input("Patient Status(ACTIVE/RECOVERED/DECEASED):").upper()
            #Replace 'ACTIVE' with entered patient status
            line=line.replace("ACTIVE",patientStatus)+"\n"
            #Renew index in fileData after modifying patient's status
            fileData[index]=line
            print(line)
    fileHandler.close()
    file=open('Patient_Status.txt','w')
    #This is a for loop to run through each line in fileData for writing line into file
    for line in fileData:
        file.write(line)
    file.close()

#------------------------------------------------------------------------------------------------------------------------------------------------

#Total number of patients in each test
def testCarriedOut():
    #All counts start from 0
    count1=0
    count2=0
    count3=0
    fileHandler = open('Test1_negative.txt','r')
    #This is a for loop to run through each item in fileHandler
    for items in fileHandler:
        #Each line represents 1 patient
        count1 += 1
    fileHandler.close()
    fileHandler = open('Test1_positive.txt','r')
    for items in fileHandler:
        count1 += 1
    fileHandler.close()
    fileHandler = open('Test2_negative.txt','r')
    for items in fileHandler:
        count2+=1
    fileHandler.close()
    fileHandler = open('Test2_positive.txt','r')
    for items in fileHandler:
        count2+=1
    fileHandler.close()
    fileHandler = open('Test3_negative.txt','r')
    for items in fileHandler:
        count3+=1
    fileHandler.close()
    fileHandler = open('Test3_positive.txt','r')
    for items in fileHandler:
        count3+=1
    fileHandler.close()
    print("Total number of Test1 carried out is",count1)
    print("Total number of Test2 carried out is",count2)
    print("Total number of Test3 carried out is",count3)

#Total number of tested patients
def patientsTested():
    #Count starts from 0
    count=0
    fileHandler = open('Test1_negative.txt','r')
    #This is a for loop to run through each item in fileHandler
    for items in fileHandler:
        #Each line represents 1 patient
        count += 1
    fileHandler.close()
    fileHandler = open('Test1_positive.txt','r')
    for items in fileHandler:
        count += 1
    fileHandler.close()
    print("Total number of patients tested is",count)

#Total number of recovered patients
def recoveredCases():
    count=0
    status="RECOVERED"
    fileHandler = open('Patient_Status.txt','r')
    #This is a for loop to run through each item in fileHandler
    for items in fileHandler:
        if status in items:
            count+=1
    fileHandler.close()
    print("Total number of recovered cases is",count)

#Total number of positive patients in each group
def positiveGroup():
    #All counts start from 0
    count1=0
    count2=0
    count3=0
    count4=0
    count5=0
    fileHandler = open('Test1_positive.txt','r')
    #This is a for loop to run through each item in fileHandler
    for item in fileHandler:
        #First 5 alphabet is the patient ID, and alphabet 1-3 is the abbreviations of group
        if item[1:4]=='ATO':
            count1+=1
        elif item[1:4]=='ACC':
            count2+=1
        elif item[1:4]=='AEO':
            count3+=1
        elif item[1:4]=='SID':
            count4+=1
        elif item[1:4]=='AHS':
            count5+=1
    fileHandler.close()
    fileHandler = open('Test2_positive.txt','r')
    for item in fileHandler:
        if item[1:4]=='ATO':
            count1+=1
        elif item[1:4]=='ACC':
            count2+=1
        elif item[1:4]=='AEO':
            count3+=1
        elif item[1:4]=='SID':
            count4+=1
        elif item[1:4]=='AHS':
            count5+=1
    fileHandler.close()
    fileHandler = open('Test3_positive.txt','r')
    for item in fileHandler:
        if item[1:4]=='ATO':
            count1+=1
        elif item[1:4]=='ACC':
            count2+=1
        elif item[1:4]=='AEO':
            count3+=1
        elif item[1:4]=='SID':
            count4+=1
        elif item[1:4]=='AHS':
            count5+=1
    fileHandler.close()
    print("Total number of positive patients in ATO",count1)
    print("Total number of positive patients in ACC",count2)
    print("Total number of positive patients in AEO",count3)
    print("Total number of positive patients in SID",count4)
    print("Total number of positive patients in AHS",count5)

#Total number of positive patients in each zone
def positiveZone():
    #All counts start from 0
    count1=0
    count2=0
    count3=0
    count4=0
    fileHandler = open('Test1_positive.txt','r')
    #This is a for loop to run through each item in fileHandler
    for item in fileHandler:
        #First 5 alphabet is the patient ID, and alphabet 0 is the abbreviations of zone
        if item[0]=='A':
            count1+=1
        elif item[0]=='B':
            count2+=1
        elif item[0]=='C':
            count3+=1
        elif item[0]=='D':
            count4+=1
    fileHandler.close()
    fileHandler = open('Test2_positive.txt','r')
    for item in fileHandler:
        if item[0]=='A':
            count1+=1
        elif item[0]=='B':
            count2+=1
        elif item[0]=='C':
            count3+=1
        elif item[0]=='D':
            count4+=1
    fileHandler.close()
    fileHandler = open('Test3_positive.txt','r')
    for item in fileHandler:
        if item[0]=='A':
            count1+=1
        elif item[0]=='B':
            count2+=1
        elif item[0]=='C':
            count3+=1
        elif item[0]=='D':
            count4+=1
    fileHandler.close()
    print("Total number of positive patients in Zone A",count1)
    print("Total number of positive patients in Zone B",count2)
    print("Total number of positive patients in Zone C",count3)
    print("Total number of positive patients in Zone D",count4)

#Statistical information on tests carried out
def statisticalInformation():
    choice=0
    while (choice!=6):
        print('Total number of')
        print('1. Tests carried out')
        print('2. Patients tested')
        print('3. Recovered cases')
        print('4. Patients test positive for COVID-19 group wise')
        print('5. Active cases zone wise')
        print('6. Exit')
        choice=input('Enter selection: ')
        #'choice' must be a number(1,2,3,4,5,6)
        try:
            choice=int(choice)
            if choice==1:
                testCarriedOut()
            elif choice==2:
                patientsTested()
            elif choice==3:
                recoveredCases()
            elif choice==4:
                positiveGroup()
            elif choice==5:
                positiveZone()
            elif choice==6:
                break
            #If 'choice' is not a number between 1-6, it will go to 'else'
            else:
                print('Invalid input')
        #If 'choice' is not a number, it will go to 'except'
        except:
            print('Non-numeric value entered.')
        print()

#-----------------------------------------------------------------------------------------------------------------------------------------------

#Information of registered patient data
def searchPatientRecord():
    try:
        fileHandler = open('Patient_Detail.txt','r')
    except:
        print ('File cannot be opened:')
        exit()
    search_key = input('Enter patient ID or name: ')
    #This is a for loop to run through each line in fileHandler
    for line in fileHandler:
        line = line.rstrip()
        #If 'search_key'(in upper case) is not in the line, it will go to next line and find 'search_key'
        if search_key.upper() in line:
            print(line)
            #When 'search_key' is found, program will print the information and return to function searchPatientData()
            return
    print("Data not found.")
    print()
    fileHandler.close()

#Information of COVID-19 positive result patient data
def searchCaseStatus():
    try:
        fileHandler = open('Patient_Status.txt','r')
    except:
        print ('File cannot be opened:')
        exit()
    search_key = input('Enter case ID: ')
    #This is a for loop to run through each line in fileHandler
    for line in fileHandler:
        line = line.rstrip()
        #If 'search_key'(in upper case) is not in the line, it will go to next line and find 'search_key'
        if search_key.upper() in line:
            print(line)
            #When 'search_key' is found, program will print the information and return to function searchPatientData()
            return
    print("Data not found.")
    print()
    fileHandler.close()

#Information of deceased patients
def deceasedPatient():
    try:
        fileHandler = open('Patient_Status.txt','r')
    except:
        print ('File cannot be opened:')
        exit()
    search_key = 'DECEASED'
    #This is a for loop to run through each line in fileHandler
    for line in fileHandler:
        line = line.rstrip()
        #If 'search_key' is not in the line, it will go to next line and find 'search_key'
        if search_key in line:
            print(line)
    print()
    fileHandler.close()

#Choice for user to search each particular patient data
def searchPatientData():
    choice=0
    while (choice!=4):
        print('1. Patient Record')
        print('2. Status of Case')
        print('3. Patient Record of all Deceased Patients')
        print('4. Exit')
        choice=input('Enter selection: ')
        #'choice' must be a number(1,2,3,4)
        try:
            choice=int(choice)
            if choice==1:
                searchPatientRecord()
            elif choice==2:
                searchCaseStatus()
            elif choice==3:
                deceasedPatient()
            elif choice==4:
                break
            #If 'choice' is not a number between 1-4, it will go to 'else'
            else:
```

```
            print('Invalid input')
        #If 'choice' is not a number, it will go to 'except'
        except:
            print('Non-numeric value entered.')
        print()

#----------------------------------------------------------------------------------------------------------------------------------------

#Create file for storing information
def createFile():
    #Store registered patient information
    fileHandler=open("Patient_Detail.txt",'a')
    fileHandler.close()
    #Store negative Test1 result patient information
    fileHandler=open('Test1_negative.txt','a')
    fileHandler.close()
    #Store positive Test1 result patient information
    fileHandler=open('Test1_positive.txt','a')
    fileHandler.close()
    #Store negative Test2 result patient information
    fileHandler=open('Test2_negative.txt','a')
    fileHandler.close()
    #Store positive Test2 result patient information
    fileHandler=open('Test2_positive.txt','a')
    fileHandler.close()
    #Store negative Test3 result patient information
    fileHandler=open('Test3_negative.txt','a')
    fileHandler.close()
    #Store positive Test3 result patient information
    fileHandler=open('Test3_positive.txt','a')
    fileHandler.close()
    #Store patient information who test positive for COVID-19
    fileHandler=open("Patient_Status.txt",'a')
    fileHandler.close()

#Main Menu
def menu():
    choice=0
    while choice!=6:
        print('----------COVID-19 Patient Management System----------')
        print('Select the operation that you want to perform:')
        print('1. New Patient Registration')
        print('2. Test Result and Action Taken')
        print('3. Changing Patient Status')
        print('4. Statistical Information on Tests Carried Out')
        print('5. Searching Functionalities')
        print("6. Exit")
        choice =input('Enter selection: ')
        print()
        createFile()
        #'choice' must be a number(1,2,3,4,5,6)
        try:
            choice=int(choice)
            if choice==1:
                patientRegistration()
            elif choice==2:
                testResults()
            elif choice==3:
                modifyPatientStatus()
            elif choice==4:
                statisticalInformation()
            elif choice==5:
                searchPatientData()
            elif choice==6:
                break
             #If 'choice' is not a number between 1-6, it will go to 'else'
            else:
                print('Invalid input')
        #If 'choice' is not a number, it will go to 'except'
        except:
            print("Non-numeric value entered.")
        print()

menu()
```

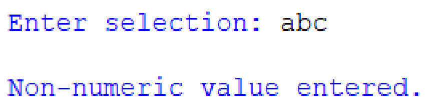
4.0 Input and Output of Program

4.1 Main Menu

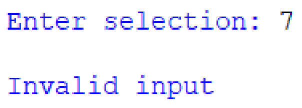


```
----------COVID-19 Patient Management System----------
Select the operation that you want to perform:
1. New Patient Registration
2. Test Result and Action Taken
3. Changing Patient Status
4. Statistical Information on Tests Carried Out
5. Searching Functionalities
6. Exit
Enter selection:
```

When users run the program, the first output is the main menu of the program. There are five selections provided for users to choose either one according to their requirements. Each selection has its specific function. Besides integers from 1 to 6, users are not allowed to enter other numbers or alphabets.

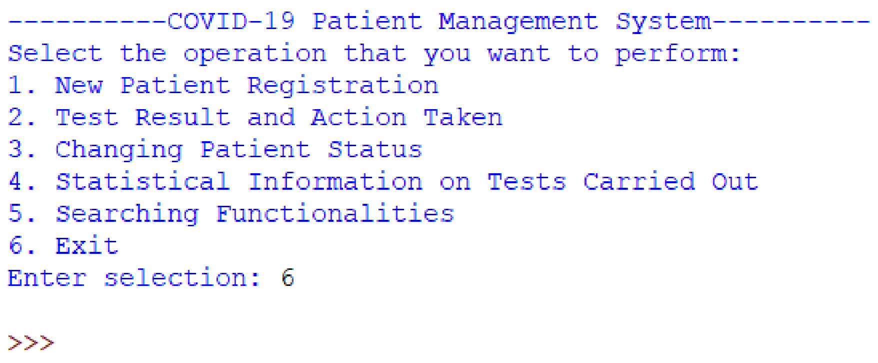


```
Enter selection: abc

Non-numeric value entered.
```



```
Enter selection: 7

Invalid input
```

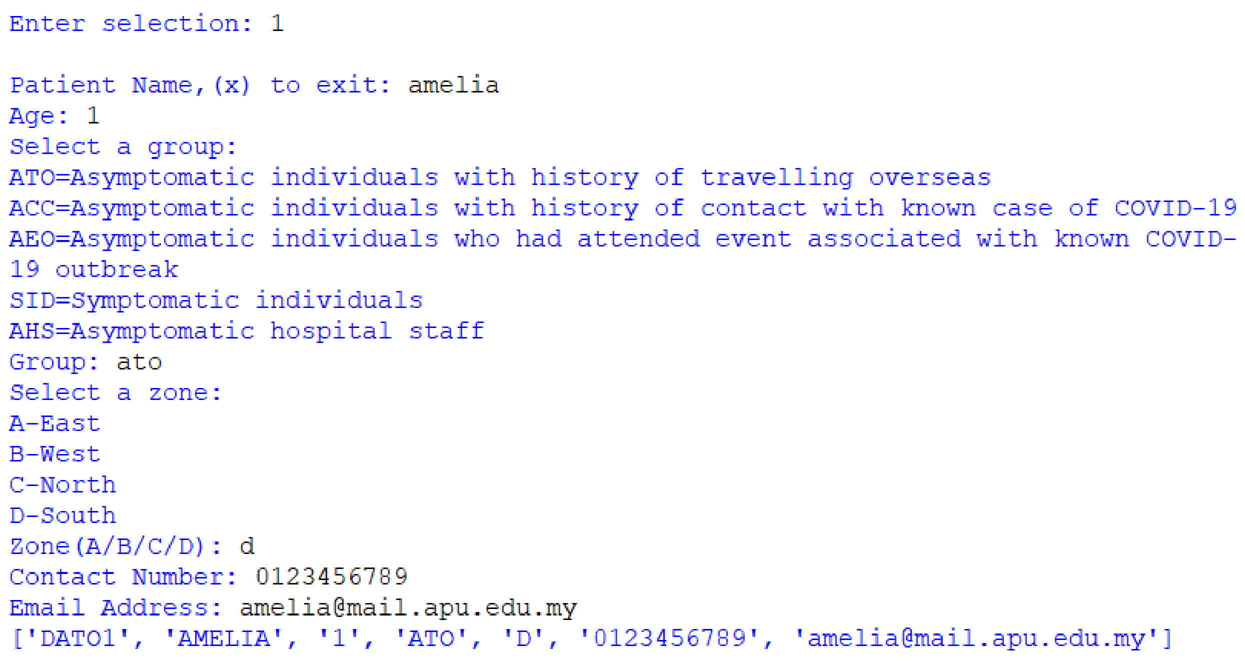
It will show "Non-numeric value entered" when the entered input is not a number. Besides, it will show "Invalid input" when the entered number is not between 1 to 6.



```
----------COVID-19 Patient Management System----------
Select the operation that you want to perform:
1. New Patient Registration
2. Test Result and Action Taken
3. Changing Patient Status
4. Statistical Information on Tests Carried Out
5. Searching Functionalities
6. Exit
Enter selection: 6

>>>
```

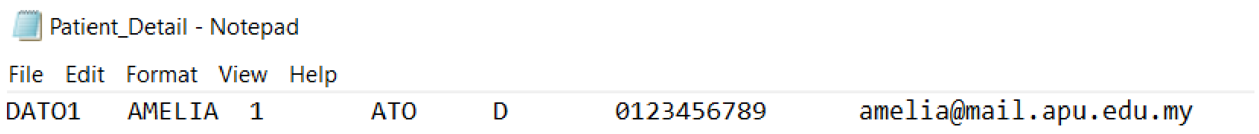
If users enter 6, the program will end automatically.

4.2 Patient Registration

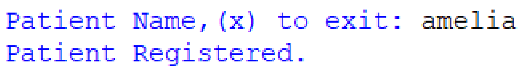


```
Enter selection: 1

Patient Name,(x) to exit: amelia
Age: 1
Select a group:
ATO=Asymptomatic individuals with history of travelling overseas
ACC=Asymptomatic individuals with history of contact with known case of COVID-19
AEO=Asymptomatic individuals who had attended event associated with known COVID-
19 outbreak
SID=Symptomatic individuals
AHS=Asymptomatic hospital staff
Group: ato
Select a zone:
A-East
B-West
C-North
D-South
Zone(A/B/C/D): d
Contact Number: 0123456789
Email Address: amelia@mail.apu.edu.my
['DATO1', 'AMELIA', '1', 'ATO', 'D', '0123456789', 'amelia@mail.apu.edu.my']
```

If user entered 1 for selection, the program provides users to enter the information of new patient. The above picture shows the process of patient registration. The entered information will save in “Patient_Detail.txt” file as the figure shown below. In patient registration, the age and contact number must be a number, the group must be one of the groups provided (ATO, ACC, AEO, SID, AHS) and the zone must be one of the zones provided (A, B ,C, D). The patients’ ID will be automatically generated by the program. Furthermore, name, group and zone of patients will be recorded in upper case .



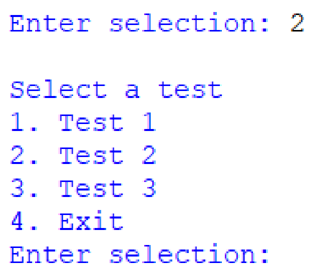
```
DATO1    AMELIA   1        ATO      D        0123456789       amelia@mail.apu.edu.my
```



```
Patient Name,(x) to exit: amelia
Patient Registered.
```

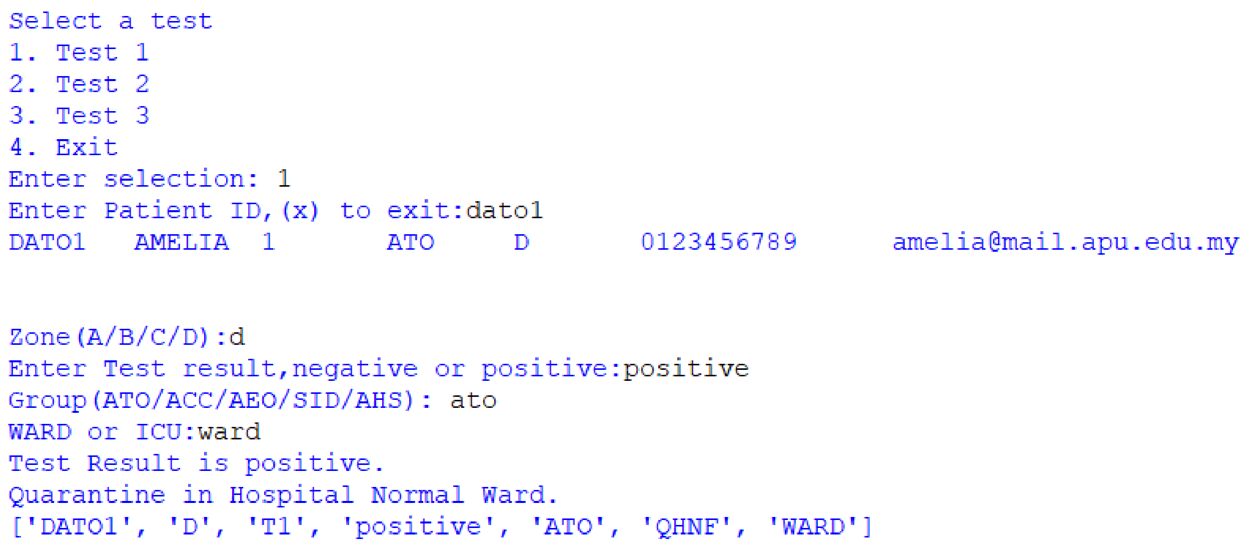
Patients who have completed registration before are not allowed to register again.

4.3 Test Result and Action Taken



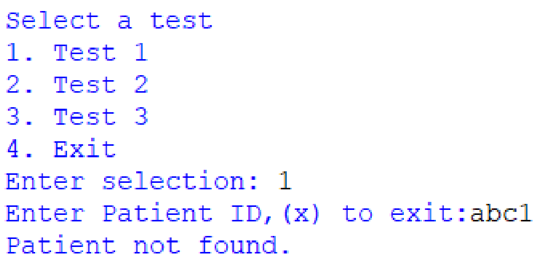
```
Enter selection: 2

Select a test
1. Test 1
2. Test 2
3. Test 3
4. Exit
Enter selection:
```

Selection 2 is a function of recording patients' test result and action taken. It provides users to choose either one of the tests. It is important that patients must go through the test1 or they can not run the following tests. Same as the main menu, users are not allowed to enter alphabet and numbers besides from 1 to 4.

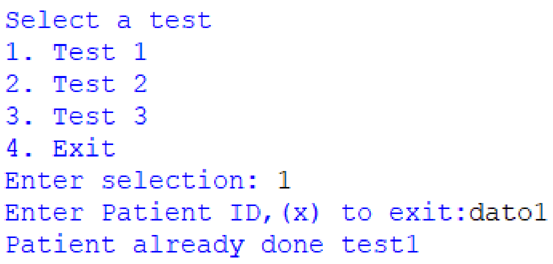


```
Select a test
1. Test 1
2. Test 2
3. Test 3
4. Exit
Enter selection: 1
Enter Patient ID,(x) to exit:dato1
DATO1    AMELIA   1       ATO     D       0123456789      amelia@mail.apu.edu.my


Zone(A/B/C/D):d
Enter Test result,negative or positive:positive
Group(ATO/ACC/AEO/SID/AHS): ato
WARD or ICU:ward
Test Result is positive.
Quarantine in Hospital Normal Ward.
['DATO1', 'D', 'T1', 'positive', 'ATO', 'QHNF', 'WARD']
```

The diagram above shows the process of entering test result of a patient. Beside from test result, all the information will be recorded in upper case. In this function, group and zone must be in the range of provided groups and zones, test result must be either negative or positive. Patients have to choose either normal ward or ICU according to each patient's situation.



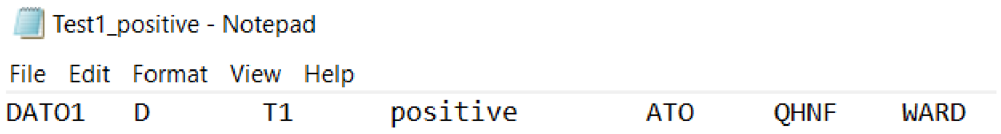
```
Select a test
1. Test 1
2. Test 2
3. Test 3
4. Exit
Enter selection: 1
Enter Patient ID,(x) to exit:abc1
Patient not found.
```



```
Select a test
1. Test 1
2. Test 2
3. Test 3
4. Exit
Enter selection: 1
Enter Patient ID,(x) to exit:dato1
Patient already done test1
```

There are two situation for patients who are not available to run the test:

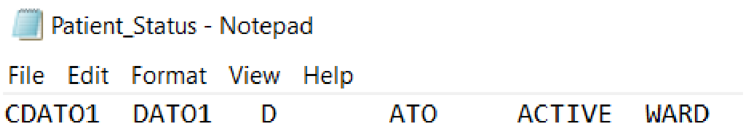
1. Patients who didn't complete registration cannot run test1. If patients didn't complete test1 and their previous test result is positive, they cannot run the following test.
2. Patients who already done the particular test, they are not required to run the test again. For example, patients who have done test1 are not allowed to go through test1 again.



```
DATO1    D        T1        positive        ATO      QHNF      WARD
```

In each test, if the patients' test result is positive, their information will automatically record in each test positive text file. If the patients' test result is negative, their information will automatically record in each test negative text file.

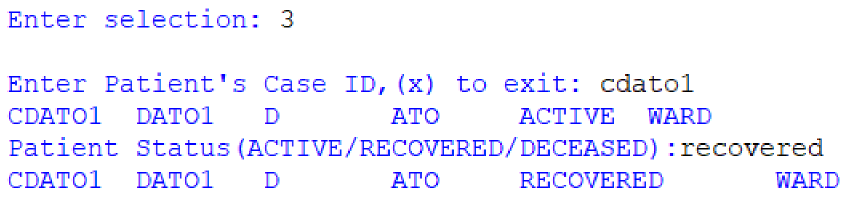
4.4 Modifying patients’ status


Patient_Status - Notepad

File Edit Format View Help

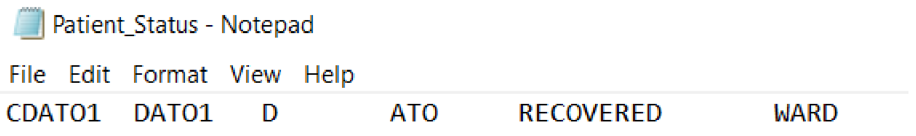
```
CDATO1   DATO1    D         ATO       ACTIVE   WARD
```

In the previous program, when the patients’ test result is positive, the program will automatically generate information in “Patient_Status.txt” file for recording the active COVID-19 patients’ information and their status.



```
Enter selection: 3

Enter Patient's Case ID,(x) to exit: cdato1
CDATO1   DATO1    D         ATO       ACTIVE   WARD
Patient Status(ACTIVE/RECOVERED/DECEASED):recovered
CDATO1   DATO1    D         ATO       RECOVERED        WARD
```

The third selection in menu provides the function of modifying the patients’ status. The diagram shown above is the process of modifying patients’ status. It is crucial that the entered patient’s case ID must be existing in “Patient_Status.txt” file and the patient’s status must be “ACTIVE”. Recovered and deceased patients’ status cannot be changed.

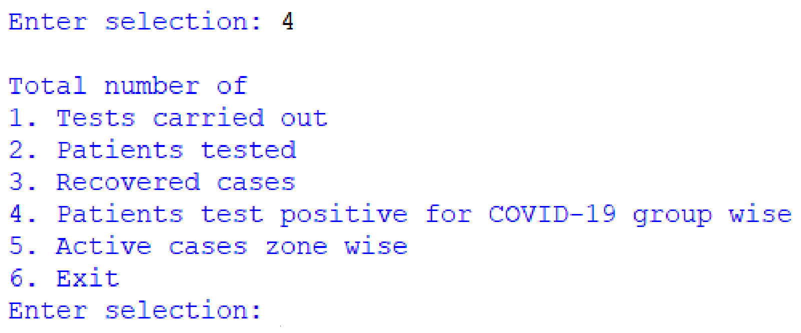

Patient_Status - Notepad

File Edit Format View Help

```
CDATO1   DATO1    D         ATO       RECOVERED        WARD
```

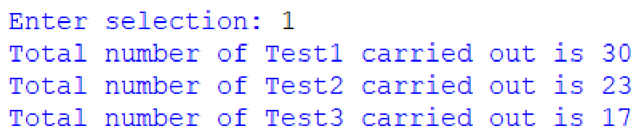
After changing the patients’ status, the information in “Patient_Status.txt” file will be changed simultaneously.

4.5 Statistical Information

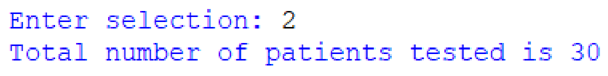


```
Enter selection: 4

Total number of
1. Tests carried out
2. Patients tested
3. Recovered cases
4. Patients test positive for COVID-19 group wise
5. Active cases zone wise
6. Exit
Enter selection:
```

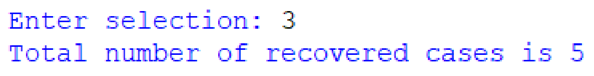
The 4th selection in menu allows users to check the total number of selected group of patients. Users are able to check the statistical information of total number of tests carried out, tested patients, recovered cases and positive patients for COVID-19 in each groups or zones.



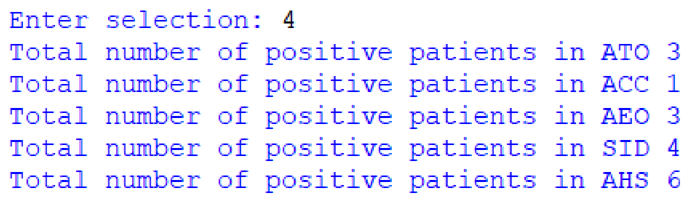
```
Enter selection: 1
Total number of Test1 carried out is 30
Total number of Test2 carried out is 23
Total number of Test3 carried out is 17
```



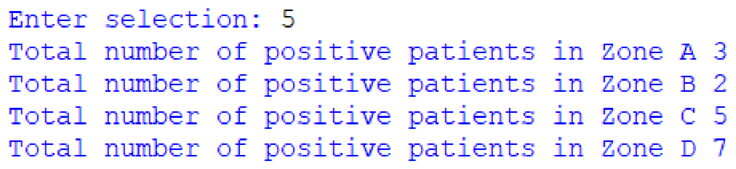
```
Enter selection: 2
Total number of patients tested is 30
```



```
Enter selection: 3
Total number of recovered cases is 5
```



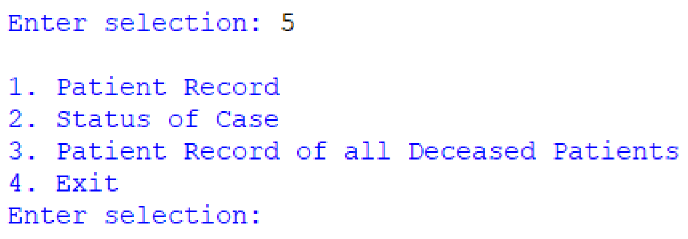
```
Enter selection: 4
Total number of positive patients in ATO 3
Total number of positive patients in ACC 1
Total number of positive patients in AEO 3
Total number of positive patients in SID 4
Total number of positive patients in AHS 6
```



```
Enter selection: 5
Total number of positive patients in Zone A 3
Total number of positive patients in Zone B 2
Total number of positive patients in Zone C 5
Total number of positive patients in Zone D 7
```

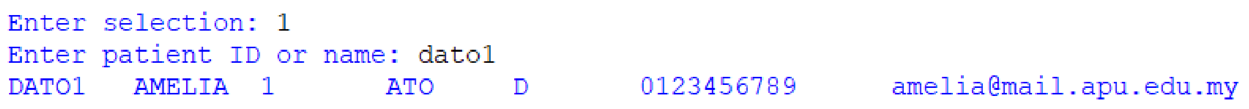
Five diagrams shown above shows that the statistical information of the designed program.

4.6 Searching Functionalities

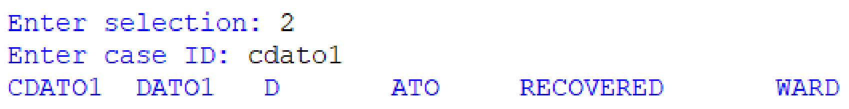


```
Enter selection: 5

1. Patient Record
2. Status of Case
3. Patient Record of all Deceased Patients
4. Exit
Enter selection:
```

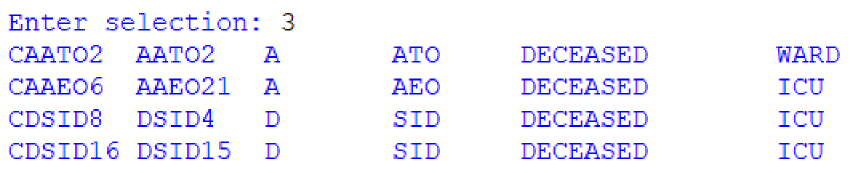
The last function of this program provides the users to search the patients or cases they wanted.



```
Enter selection: 1
Enter patient ID or name: dato1
DATO1   AMELIA  1       ATO     D       0123456789      amelia@mail.apu.edu.my
```



```
Enter selection: 2
Enter case ID: cdato1
CDATO1  DATO1   D       ATO     RECOVERED       WARD
```



```
Enter selection: 3
CAATO2  AATO2   A       ATO     DECEASED        WARD
CAAEO6  AAEO21  A       AEO     DECEASED        ICU
CDSID8  DSID4   D       SID     DECEASED        ICU
CDSID16 DSID15  D       SID     DECEASED        ICU
```

The first selection in searching functionalities allows users to search patients according to their names or patients’ ID. The second selection provides users to search the positive for COVID-19 case according to the case ID. The last functionality will print out the information of all the deceased patients.

5.0 Conclusion

COVID-19 Patient Management System allows organizations or hospitals to record the information of patients. It is user friendly and suitable for any users who know English. Compared to other applications, the designed program is simpler.

In the code, the program was written in Python programming language. It is menu-driven. The program is completed using basic python programming knowledge such as variables, string, file handling, and others. Besides, list and function technique are used in this program.

I have learnt the basic knowledge of programming while completing the assignment. Defining the problems and translating them into programmable solutions using flowchart and pseudocode is the first thing needed to be done in every coding program. Python programming language provide users readability and it is more understandable for the beginners.