Hacettepe University

Department of Computer Engineering

BBM 103

Assignment 2: Doctor’s Aid

2210356102- Yunus Emre Uluer

25.11.2022

Analysis

The problem that we were given was about diagnoses and mathematical probabilities of cancer. We were given a doctors_aid_input.txt file and we were asked for an output file that gives the results of the input file.

In the input file, 5 commands provide different functions. Each line identifies a name(commands) and the parameters of that command. Although the parameters of that command may not be in the list in our program. So, this should give an output that informs the user that the function can not calculate it due to absence.

Our program should read and write sequentially as the commands are executed. This cause (when you create patients after you asked for their calculations) a problem because commands are being executed sequentially.

Create command takes the patient's name, diagnosis accuracy, disease name, disease incidence, treatment name, and treatment risk in order. And it should record it.

Remove command takes patients' names and removes their data from the program. If there is no patient on the list, it should do nothing and inform the user about it.

The list command takes the list of the patients and makes a chart that can be used by the doctors. Aligning the columns is an important aspect of this function.

Probability command is about the diagnosis accuracy and the incidence of the disease. Our patients are the people whose cancer tests are positive. But because of the real incidence and the accuracy of that test, our patients may not be cancer. This probability can be calculated by dividing the true positive patients by the true and false positive patients. Calculations will be discussed in the programmer's catalogue later.

Recommendation command takes the probability of the patient and compares it with the risk of that treatment. If it is bigger than the risk function gives an output that suggests the treatment. Otherwise, it does not suggest it. Also, like every command in this program it should inform the user when the patient's name is not in the list

Patients' lists should be held in a multidimensional array. And the first element of the lists of the patient list should be the patient's name.

Design

This problem should be solved by subproblems(functions) and a multidimensional array. Readlines function gives a list that the element of the lists are lines. But the data in the list contains lots of special characters like \t and \n. str.split() function can be used for creating the data list. After creating the list, we can look for the parameters (patient’s names) in the array (first elements of the lists). Removing the command can make it better for searching the name. After handling the array, we can create the functions we need. For the output of the commands, we can create a function named write() , which can behave like a print function but writes the string in the output file.

Reading Data:

Os functions will be helpful with reading the data from text file. And we are supposed to make it a function. Functions open it and make it array splitting with the ‘\n’ character.

Writing Data :

This function should be used in nearly every function in the program and takes parameters and add it to the output file.

Sorting Data:

Writing a function that checks every element in the data array and str.split() it and removes the command and makes it a list but from the parameters of creating there are ',' and ‘ ‘ in the string so it should be removed too. Also in the input file risks and accuracies are given like 0.9999 and it should be transformed into a 99.99%. We should return the list to the create command so it can append it to the data list.

Commands :

The program should check which command it is and forward it to the related function. This function checks the first array in the list and forwards it to the related part after calling sorting data function.

Create :

This function takes the returned list and appends it. Also checks whether the patient is in the list or not. We should write it if the patient's name is on the list.

Remove :

Takes the patient's name and removes the list of them. If there is no, inform the user with write function

Probability :

Uses the second and fourth elements from the data of the patient and calculates the probability. Uses write functions to inform. Also, return the probability value to the recommendation function.

Recommendation:

Takes the value from the probability function and compares it with the risk of the patient’s treatment. Calculating the probability of the patient is provided by calling the probability function. But the probability function writes output when calculating the probability. Therefore, we can write another probability function that returns the probability but does not write it. The other solution can be also writing the probability after the probability function in the commands part.

Pseudocode of the program :

```
Define all the functions above and make a list that has no element.
Call it list_of_data.
Read the input file for each element in array :
    checkcommands(element)
checkcommands(element) looks what data with searching the element of the element :
    Example : if element[0] == create :
        Sort it and create()
def create() = At the end of the create function write( x, y ,z )
(x , y , z) is the string that the program supposed to write.
Check for each element (each line) sequentially and end the loop.
```

Programmer’s Catalogue

Probability Calculations	I read the document about the confusion matrix, but it took time for me to fully understand and implement it correctly. After 45 minutes, the problem was solved.
OS functions	I did not know about the module so; I first checked the resources. Then I made some research about this module and then decided to go with the documents we are given. This part took 1 hour.
Design	Designing the algorithm was a bit easier than the other parts. I thought about it for 30 minutes and almost everything was clear in my mind. The tricky point of the design was the usage of arrays. At first, I was going to create 2-3 different arrays and connect them. But I decided to go with the one array. This brought some advantages and disadvantages. All of them took me approximately 2 hours to design.
Handling errors/Implementing	Implementing it took 4-5 hours. Because I divided the problems into sub-solutions, I was handling errors as I write codes. After 5-6 hours after the design part, the program was done.
Testing	Testing was the most difficult thing in the process of this assignment. I had a hard time aligning the columns. I could not find what is missing so I lost lots of time on testing. It took about 1.5 hours.
Reporting	I wrote my design before I started to implement it. For this reason, I changed most of the parts in the design part and writing it many times was the reason for the time spent. It took 5 hours to write the report.

Source code: (And their explanations)

```
--------------------------------------------------------------------------------


import os

current_dir_path = os.getcwd()
reading_file_name = "doctors_aid_inputs.txt"
reading_file_path = os.path.join(current_dir_path, reading_file_name)

for



writing_file_name = "doctors_aid_outputs.txt"
writing_file_path = os.path.join(current_dir_path, writing_file_name)
```

This part of the code reads paths of txt’s (writing and reading file)

```
--------------------------------------------------------------------------------
with open (writing_file_path , 'w') as f :
    f.write('')
```

This part is for overwriting for doctors_aid_outputs.txt. Writes nothing

```
--------------------------------------------------------------------------------

Read function:

def read():
    with open(reading_file_path, "r") as f:
        global data
        data = f.readlines()
```

Read function reads data and makes a list.

Sort data function:

```
def sort_data(sortdata) :
    #getting rid of the commands
    sortdata = str.split(sortdata)
    sortdata.remove(sortdata[0])
    #checking if command is create or not
    if len(sortdata) > 5 :
        #getting rid of the ','s
        sortdata =' '.join(sortdata)
        sortdata = sortdata.split(',')
        #gettin rid of the whitespaces
        for i in range(2,5):
            a = list(sortdata[i])
            a.remove(" ")
            sortdata[i]=''.join(a)

        #arranging accuracy
        float_accuracy = float(sortdata[1])
        string_accuracy =  str("{0:.2f}".format(float_accuracy*100)) + '%'
        sortdata[1] = string_accuracy
        #arranging risk
        float_risk = float(sortdata[5])
        string_risk = str(int(float_risk*100)) + '%'
        sortdata[5] = string_risk


    return sortdata
```

Sort data function takes the parameter from create_commands() function and split it with str.split and removes the command (only the [Name] part remains).And if it is a create function(which has 7 parameters) it sorts it by getting rid of the ‘,’ and ‘ ’. Adding % etc. It returns the value for create() function

----

```python
def write(x,y,z) :
    with open (writing_file_path , 'a') as f :
        x = x + y + z
        f.write(x)
```

Takes 3 parameters by some functions and then writes it. Behaves like print function but it writes on a txt file

----

```python
def writing_probability(patient) :
    #list to string
    patient = patient[0]
    for i in range(len(list_of_data)) :
        #probability calculations
        if patient in list_of_data[i] :
            a = list(list_of_data[i][1])
            a.remove('%')
            b = ''.join(a)
            accuracy = float(b)

            splitted_incidence = list_of_data[i][3].split('/')
            nominator = float(splitted_incidence[0])
            denominator = float(splitted_incidence[1])


            TP = (nominator*accuracy)/100
            FP = (denominator-nominator)*(100-accuracy)/100
            probability  = (TP/(TP+FP))*100
            probability = str("{0:.4g}".format(probability)) + "%"
            x = "Patient " + list_of_data[i][0]
            y = " has a probability of " + probability
            z = " of having " + list_of_data[i][2].lower() +".\n"
            write(x,y,z)
            return
    write("Probability for ", patient , " cannot be calculated due to 
absence.\n")
```

This function calculates probability. Takes data from the list. (patient is the parameter of the function).removes '%' and '/' ,takes data , and calculates probability. Calculation of probability is like this: The patient's test is positive but the test can be false positive or true positive. FP can be calculated as the denominator-nominator of disease incidence and multiply it by (100-accuracy)/100. This number gives the value of false positives due to accuracy of test. TP can be calculated like that also. TP/TP+FP gives the probability. Returns no value. If there is no patient in the list it writes an error message.

```
--------------------------------------------------------------------------------


def calculating_probability(patient):
    #for returning recommendatiton
    patient = patient[0]
    for i in range(len(list_of_data)) :

        if patient in list_of_data[i] :
            a = list(list_of_data[i][1])
            a.remove('%')
            b = ''.join(a)
            accuracy = float(b)

            splitted_incidence = list_of_data[i][3].split('/')
            nominator = float(splitted_incidence[0])
            denominator = float(splitted_incidence[1])


            TP = (nominator*accuracy)/100
            FP = (denominator-nominator)*(100-accuracy)/100
            probability  = (TP/(TP+FP))*100

            return probability
```

This is almost the same function as the writing_probability but instead returns value for recommendation function.

```
--------------------------------------------------------------------------------
```

```
def create(commands) :
    #appending list and checking if there is dublication or not
    global list_of_data
    for i in list_of_data :
        if commands[0] == i[0] :
            write("Patient ", i[0] , " cannot be recorded due to duplication.\n")
            return
    write("Patient ", commands[0] , " is recorded.\n")
    list_of_data.append(commands)
```

Create function effects list of data and take parameters from sort_data command. If the patient is already in the list it writes and error message(by checking every patient) it appends patients data to the list.

```
--------------------------------------------------------------------------------

def listing() :

    text =
"Patient\tDiagnosis\tDisease\t\t\tDisease\t\tTreatment\t\tTreatment\nName\tAccura
cy\tName\t\t\tIncidence\tName\t\t\tRisk\n-----------------------------------------
---------------------------------\n"
    for i in list_of_data:
        line= ""
        #all of the if statements are written based on the longest word
        if(len(i[0]) < 3) :
            line +=  i[0] + "\t\t"
        else :
            line +=  i[0] + "\t"
        line += i[1] + "\t\t"
        if (len(i[2]) < 12) :
            line += i[2] +"\t\t"
        else:
           line+= i[2]+ "\t"
        line+=i[3]+"\t"
        if (len(i[4]) < 8) :
           line+=i[4]+"\t\t\t"
        elif (len(i[4]) < 12) :
           line+=i[4]+"\t\t"
        elif (len(i[4]) < 16) :
           line+=i[4]+"\t"

        else:
            line+=i[4]
        line+= i[5] + "\n"
        text += line
    write(text,"","")
```

Listing function writes some string and align them. Aligning was created with the usage of the given longest words. Function puts tab by 4 character long space according to the given longest word.This function does this for every element in list.

```
def removedata(commands) :
    commands= ''.join(commands)
    global list_of_data
    for i in range(len(list_of_data)-1) :
        if commands == list_of_data[i][0] :
            list_of_data.remove(list_of_data[i][: len(list_of_data[i])])
            write("Patient ",commands," is removed.\n")
            return
    write("Patient ",commands," cannot be removed due to absence.\n")
```

Behaves like create but removes it by using sublist.

```
def recommendation(patient) :
    patient = patient[0]
    for i in range(len(list_of_data)) :
        #calculating the risk and calling probability
        if patient == list_of_data[i][0] :
            a = list(list_of_data[i][5])
            a.remove('%')
            b = ''.join(a)
            risk = float(b)
            probability = calculating_probability([patient])
            if risk > probability :
                write("System suggests " , patient , " NOT to have the
treatment.\n")
```

```
            else:
                write("System suggests " , patient , " to have the treatment.\n")
            return

    write("Recommendation for ",patient," cannot be calculated due to
absence.\n")
```

Recommendation uses calculating_probability() and compares it with risk. Returns none.

```
#-------------------------------------------------------------------------------------



def check_command(commands) :
    commands = commands.split(' ')
    #splits for checking commands and makes it string againg for better sorting
    if commands[0] == 'create' :
        commands = ' '.join(commands)
        commands = sort_data(commands)
        create(commands)
    elif commands[0] =='remove' :
        commands = ' '.join(commands)
        commands = sort_data(commands)
        removedata(commands)
    elif commands[0] == 'recommendation':
        commands = ' '.join(commands)
        commands = sort_data(commands)
        recommendation(commands)
    elif commands[0] == 'probability':
        commands = ' '.join(commands)
        commands = sort_data(commands)
        writing_probability(commands)
    elif commands[0] == 'list\n' or commands[0] == 'list':
        listing()
```

check_commmand function takes unsorted list by the program and splits it first.(by space).Then, it looks for the first element of the lists(lines) and jumps to the related part. Sort data can sort it better so command part makes it string again by commands = ' '.join(commands). Functions use returned value(sort functions returned value).

```
#-------------------------------------------------------------------------------------
```

```python
read()
list_of_data = []
for i in range(len(data)) :
    check_command(data[i])
```

Main function of the program. It reads the data . For each line , it checks command.

User Catalogue

In the folder of the code , you can create a text file and give the commands. Text file's name must be doctors_aid_input and when you write the commands the order must be matched :

For recommendation/probability/remove:

(command) (patients name)

For list :

list

For create

create (patient name), (diagnosis accuracy), (disease incidence), (treatment name), (treatment risk)

This order should be preserved. But the order of commands(lines) does not matter. Though, if you search for a probability of a patient before you create it you can not get what you want. Remember, the program executes your input file sequentially.

The program cannot save your data in some files so either you keep the list in the output file or you need to change the input file every time you want the change the program.

Grading

Evaluation	Points	Evaluate Yourself / Guess Grading
Indented and Readable Codes	5	5
Using Meaningful Naming	5	5
Using Explanatory Comments	5	5
Efficiency (avoiding unnecessary actions)	5	4
Function Usage	25	25
Correctness	35	32
Report	20	17
There are several negative evaluations … …	…	-2