ASSIGNMENT REPORT

BBM103

ASSIGNMENT-2

EMRE ÇİĞİL – 2210356033

24.11.2022

ANALYSIS

In the problem, we are asked to simulate a doctor assistance program. We must register the patients whose information has been given in the system, delete them from the system and take various actions for the information.

Each patient is given information about cancer. For example, the type of cancer, the type of treatment, the risk threshold, etc. We use this information to see what type of treatment they should be treated with or interpret whether they should be treated or not. Finally, we see all the patients in a chart of lists.

We will have some conditions when creating this simulation. Some of our patients may not be involved in some commands or functions, or they may be involved. For example, if we want to register a private patient, it will not register patients other than that. We specify it to register the patient we want to register. Finally, we have commands for the operation we want to do. Using these commands we perform operations.

DESIGN

When I started the problem, I first used the "file_input" and "file_output" functions to get input from the outside and to write output outside the project. Before I started the cycles, I created a list called "total_patients". there will be patient information added to this list, and this list is a "permanent" list. After creating my list, I had the information in our txt file read line by line with a "for" loop. ".replace and . split" commands, I edited the information in the line. I cleared it of dots and gaps. here is a simple and helpful list of "list1". With this list, it puts the cancer types consisting of two words into the same element. After all the edits are made, I put the words in the line in the list with the name "input_list" respectively. Since this list is in a "for" loop, it adds the new row in each round. so I created a provisional list. I'm going to call functions with this list. I used the phrase "if" before calling functions because I noticed that the word "target therapy" is separate. They created problems in my solution because they were written separately. I solved this problem with the phrase "if". After that, I used the phrase "if and elif". Thanks to these statements, I call whatever function I want to do. When calling the function, I call the function according to the word special command in "input_list". For example, when we read the "create" command that we use to register a patient, it calls the creat function.

I wrote the functions above the solution. my first function is "create()". With this function, I add patients to my permanent list. With the if statement, I check whether our patient is on the permanent list or not. I don't need to register the patient if any. I warn this with the print() command so that the user can see it under the expression. If the patient is not on the permanent list, he adds the patient to the permanent list. I fixed the second and sixth element inside the "input_list" list before adding it to the permanent list. Because in the future patients will print out the

graph with the "list" function and the graph should be more understandable.

My second function is "remove()". With this function, I can delete patients from the permanent list. To delete it, I first use a for loop because the patient I want to delete needs to search for it in the permanent list. With the if statement, we delete the patient, if any. The elif statement indicates that the patient is not on the list. I used the variable x in this loop. Because if we don't find the patient in the cycle, the "i" variable will go back to the beginning.

My third function is "show_list()". With this function, I can see the patients on my permanent list and their information in the "output" file. In this function I calculated the distance between the headings. I created and used an algorithm for the distance between patients and their information, trying it over and over again. The logic of the algorithm is as follows: The distance between the permanent list elements depends on how many letters the elements consist of. I create distance according to the number of letters. For example, if the total distance between words should be 20 spaces, I subtract the number of letters of the first word from 20. It may seem complicated, but I think it's very useful. when we call this function, we can see information in the form of a diagram.

The fourth function is the "probability()" function. With this function, I calculate the probability of the patient having a disease by mathematical operation. This calculation is called the "confusion matrix". I print out the probability that the patient is sick. If the patient is not on the permanent list, I explain this with the phrase "if and elif".

The fifth function is the "recommendation()" function. With this function, I calculate the probability of the patient with the "confusion matrix" and then make a recommendation according to the threshold value. For example, if the patient's probability is greater than the threshold, the system recommends that he or

she have chemotherapy. If the threshold is less than the value, the system indicates that there is no need for treatment. Here again I used the "x" value in the "for" loop. Because if he cannot find the patient's name in the permanent patient list in the cycle, he will always go back to the beginning. If the patient is not on the permanent list, the system indicates this.

I used file_output function to create an outputs file. Through this function we see everything that the user needs to see. After my research, I found that I could use this function in a shorter way. I can print this function briefly at the end of the command I will print. Normally my project was about 150-175 lines. But thanks to this method, my project became approximately 90 lines without space line. There are 7 functions in total.

PROGRAMMER'S CATALOGUE

I spent the most time on problem analysis. At first, I didn't know what the problem was and how I was supposed to solve it. I didn't know which lists and functions I should use. I spent about 4 days for analysis. I was constantly having new ideas when I was designing. And it all made sense. I chose one of them and started to design it according to the idea I chose. I spent about 1-2 days for the design. The implementation part didn't take up much of my time but I made a lot of mistakes. I spent about 1.5 days. I spent 1 day writing a report because I had never written a report before.

```
file_input = open("doctors_aid_inputs.txt", "r",
encoding="UTF-8")
file_output = open("doctors_aid_outputs.txt", "w",
encoding="UTF-8")
```

The task of these functions is to read the entered solutions and write the output after reading.

I used the "file_output" function in a different way. This reduced the number of rows and increased the efficiency of the code. It is slightly different from the usual function calling technique. When using the "Print" command, we call this function in the last part inside the parenthesis.

I want to reiterate. There are seven functions in the solution, not six.

```
if input_list[0] == "create":
    create()
elif input_list[0] == "remove":
    remove()
elif input_list[0] == "list
    show_list()
elif input_list[0] == "probability":
    probability()
elif input_list[0] == "recommendation":
    recommendation()
else:
    print("You typed an invalid command.",file=file_output)
```

The custom commands used in this "for" loop are selected individually. The special function is called according to the command. If the command is entered without an invalid command, it sends feedback to the user.

```
def create():
    if input_list[1::] in total_patients:
        print("Patient " + input_list[1] + " cannot be recorded due to
duplication.", file=file_output)
    else:
        input_list[2] = str(str("{:.2%}".format(float(input_list[2]))))
        input_list[6] = str(str(int(100 * float(input_list[6]))) + "%")
        total_patients.append(input_list[1::])
print("Patient " + input_list[1] + " is recorded.", file=file_output)
```

With the above function, we create a new patient record. After entering the information correctly, patients are added to the permanent list. The permanent list is called "total_patients". If the patient is already registered in the system, feedback is sent.

```
def remove():
    x = 0
    for i in range(len(total_patients)):
        x += 1
        if input_list[1] == total_patients[i][0]:                    # 1
check the patient name for removing by input_list and total_patient list.
            total_patients.remove(total_patients[i])                 # 1
used x for loop don't come back when searching is finish.
            print("Patient " + input_list[1]+" is removed.",
file=file_output)
            break
        elif x == len(total_patients):
            print("Patient " + input_list[1]+"  cannot be removed due to
absence.", file=file_output)
```

Through this function, the registered patient is deleted. With the for loop, the whole list is checked, and if the patient is in the list, the system deletes the patient. If the value x in the for loop reaches the final value and cannot find the patient, it sends feedback to the user.

```
def show_list():
    print("Patient Diagnosis   Disease         Disease     Treatment
Treatment", file=file_output)
    print("Name    Accuracy    Name            Incidence   Name
Risk", file=file_output)
    for i in range(72):
        print("-", end="", file=file_output)
        if i == 71:
            print("-", sep="\n", file=file_output)
    for i in range(len(total_patients)):
        for a in total_patients[i]:
            if a == total_patients[i][0]:
                space = 8 - len(a)
            elif a == total_patients[i][1]:
                space = 12 - len(a)
            elif a == total_patients[i][2]:
                space = 16 - len(a)
            elif a == total_patients[i][3]:
                space = 12 - len(a)
            elif a == total_patients[i][4]:
                space = 16 - len(a)
            elif a == total_patients[i][5]:
                space = 8 - len(a)
            print(a, end=space*" ", file=file_output)
        print(sep="\n", file=file_output)
```

Through this function, patients are shown in the form of a table. We reach all registered patients by calling this function. The headers of the table are created with the two print commands in the first row. The lines of the table are calculated with the for loop just below it. With the third for loop, we reach the sublists within the main list. with the last for loop we reach the elements in the sublist. I calculated the amount of "space" using the phrase "if" and "elif" in the For loop. So I adjusted the distance harmony between the elements in the table. For example, space = 8 - len(a). If a = "Ali", then the number of spaces between them should be 5.

```
def recommendation():
    x = 0
    for i in range(len(total_patients)):
        x += 1
        if input_list[1] == total_patients[i][0]:
            a = float((total_patients[i][3][0:2])) /
float((total_patients[i][3][3::]))
            d = float(total_patients[i][1][0:4])/100
            b = 100000 * a
            c = 100000 - b
            final_value = (100*(b / (c * (1 - d) + b)))
            if final_value > float(total_patients[i][5][0:-1]):
                print("System suggests " + input_list[1]+" to have the
treatment.", file=file_output)
            else:
                print("System suggests " + input_list[1]+" NOT to have the
treatment.", file=file_output)
            break
        elif x == len(total_patients):
            print("Recommendation for " + input_list[1] + " cannot be
calculated due to absence.", file=file_output)
```

With this function, "should the patient be treated? If so, what form of treatment should it be?" With the For cycle, the patient is searched in the list. The temporary "input_list" list evaluates each command and patient individually. In this function, if the name in the temporary list is in the permanent list, the confusion matrix is calculated by mathematical operations. Here we use the terms "if and elif" to check the permanent list. We use "else" if the patient does not need to be treated. The confusion matrix decides this. The information in the modal list is used to calculate the confusion matrix and is calculated by formula.

```
def probability():
    x = 0
    for i in range(len(total_patients)):
        x += 1
        if input_list[1] == total_patients[i][0]:
            a =
float((total_patients[i][3][0:2]))/float((total_patients[i][3][3::]))
            d = float(total_patients[i][1][0:-1])/100
            b = 100000*a
            c = 100000-b
            final_value = str((100*(b/(c*(1-d)+b))))
            if final_value[3] and final_value[4] == "0":
                final_value = final_value[0:2]
            else:
                final_value = final_value[0:5]
            print("Patient " + input_list[1] + " has a probability of " +
final_value +
                  "% of having "+total_patients[i][2].lower()+".",
file=file_output)
            break
        elif x == len(total_patients):
            print("Probability for " + input_list[1] + " cannot be
calculated due to absence.", file=file_output)
```

This function uses the confusion matrix to determine what percentage of patients the patient is. Again, with the for loop and the variable "x", the patient in the temporary list is searched in the permanent list. If the patient is on the permanent list,

the probability of the patient becoming sick is calculated. The variable "final_value" is the result of the patient. The "if and else" I use to final_value deletes the trailing zeros if the final_value equals 100.00 (for example). the phrase "elif" at the bottom indicates that the patient is not on the permanent list.

USER CATALOGUE

We have some conditions to run the program. First, you should not enter user information incompletely or excessively. you must first enter the command you want to make. because the program directs commands according to the first word of the input it receives from the user. The numbers you will write for the detection and treatment of the disease should be entered correctly. Otherwise, the death of the patient may be caused. Enter each character correctly to easily find patients in the system and to check their information. Otherwise, you will have difficulty finding the patient you want to find. If you type the same command twice when registering or deleting the patient, the system will give you feedback. For example, the patient named Hayriye was already registered. There are certain restrictions on the program. If you use a command that is not registered in the system, the system will not be able to provide you with this service and will send feedback. You should go to your doctor with the output you receive from the program. They will make the final decision.

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

References

- https://stackoverflow.com/
- https://www.w3schools.com/
- https://www.geeksforgeeks.org/
- https://en.wikipedia.org/