PSEUDO-CODE

What is pseudocode? Pseudocode is a way of **expressing an algorithm without conforming to specific syntax rules.** By learning to read and write pseudocode, you can easily communicate ideas and concepts to other programmers, even though they may be using completely different languages. For this task, I realised a few sequences of pseudocode described below:

1. Import the data into a data structure that you will create:
   To import the data, I used a class named Patient which will store the information about a patient – Name, date of birth, sex , height , weight , and the patient conditions, such as Asthmatic, Smoker etc.

```
class Patient:
        Patient_Name
        DateofBirth
        Sex
        Height
        Weight
        BodyBuild
        Smoker
        Asthmatic
        NJT_NGR
        Hypertension
        RenalRT
        IleostomyColostomy
        ParenteralNutrition
```

   To call the created object, I defined a constructor for Patient class. The role of the constructor is to prepare the object to be used:

```
        __init__(self, Patient_Name, DateofBirth, Sex, Height, Weight, Smoker,
        Asthmatic, NJT_NGR, Hypertension, RenalRT, IleostomyColostomy,
        ParentalNutrition)
                self.Patient_Name = Patient_Name
                self.DateofBirth = DateofBirth
                self.Sex = Sex
                self.Height = Height
                self.Weight = Weight
                self.BodyBuild = BodyBuild
                self.Smoker = Smoker
                self.Asthmatic = Asthmatic
                self.NJT_NGR = NJT_NGR
                self.Hypertension = Hypertension
                self.RenalRT = RenalRT
                self.IleostomyColostomy = IleostomyColostomy
                self.ParenteralNutrition = ParentalNutrition
```

   To store the data from the csv. file given , I used an array of objects named Patient_List:

```
        Patient_List = []
```

   For reading the data from the csv file, I defined a method called "read_csv_file". It will help in constructing the Patient_List array.

```
read_csv_file()
    open(file, 'r') as csv_file:
        csv_reader = csv.reader(csv_file)
        header = next(csv_file)
        if header != None:
            for each row in csv_reader:
              p = Patient (row[0] ,row[1], row[2], row[3], row[4], row[5], row[6],
              row[7], row[8], row[9], row[10], row[11])
                     add p into Patient_List
                     calculate   BMI and Age
```

2. Calculate the BMI for each patient and classify these patients as underweight, normal, overweight, or obese. Update the data structure adding the BMI

We know the BMI is calculated by this formula: **BMI = weight (Kgs) / Height^2 (m).**
To make things easier, I defined a method that will help us to calculate the BMI and classify the patients by the given criteria:

```
Calculate_BMI_and_Classification(List)
```

```
for each Patient in size_of_the_List:
       create_a_new_attritute_for_Patient_class('Classification')
       if(List[i].BodyBuild == 'Slim'):
            if(List[i].BMI < 18.5):
                List[i].Classification = 'Underweight'
            elif(List[i].BMI >= 18.5 and List[i].BMI < 25):
                List[i].Classification = 'Normal'
            elif(List[i].BMI >= 25 and List[i].BMI < 28):
                List[i].Classification = 'Overweight'
            else:
                List[i].Classification = 'Obese'

       if(List[i].BodyBuild == 'Regular'):
            if(List[i].BMI < 18.5):
                List[i].Classification = 'Underweight'
            elif(List[i].BMI >= 18.5 and List[i].BMI < 25):
                List[i].Classification = 'Normal'
            elif(List[i].BMI >= 25 and List[i].BMI < 29):
                List[i].Classification = 'Overweight'
            else:
                List[i].Classification = 'Obese'

       if(List[i].BodyBuild == 'Athletic'):
            if(List[i].BMI < 18.5):
                List[i].Classification = 'Underweight'
            elif(List[i].BMI >= 18.5 and List[i].BMI < 25):
                List[i].Classification = 'Normal'
            elif(List[i].BMI >= 25 and List[i].BMI < 30):
                List[i].Classification = 'Overweight'
            else:
                 List[i].Classification = 'Obese'
```

To see the changes that occurs, in the main method we will call this method using Patient_List as parameter.

call `Calculate_BMI_and_Classification(Patient_List)`

3. Display / Print on screen the patient name, age, BMI, and weight classification. Sort the display so that those classed as obese are shown at the top of the screen , followed by those classed as underweight, then those that are overweight with the list completed by those that are classed as normal body mass. Insert breaks every 10 patients to allow the user of the system to study the results.

To print the details of the patients and to sort the display, we are going to use another method called Classification, with one parameter – a List. On the first instance, this method will call method `Print_Classification`, which will create a temporary list with all the patients registered at that hospital. We are going to use the Temporary List to sort the patients by classification.

```
Print_Classification(List,classification,TemporaryList)
```

```
for each Patient in size_of_the_List:
        if(Patient.Classification == classification):
                add Patient into TemporaryList
```

- List represents our initial List (Patient_List)
- Classification represents the string for which we will sort the list ('Obese','Underweight','Normal',Underweight')
- TemporaryList represents the temporary list created in Classification method

```
Classification(List)
        Create the Temporary_List
        Call Print_Classification(List,'Obese',TemporaryList)
        Call Print_Classification(List,'Underweight',TemporaryList)
        Call Print_Classification(List,'Overweight',TemporaryList)
        Call Print_Classification(List,'Normal',TemporaryList)
        BreakLine = 0
        for each patient in TemporaryList:
        Display Patient_Name,Patient_Age,Patient.BMI,Patient.Classification
        BreakLine = BreakLine + 1
        if BreakLine = 10
                Break a line after 10 displayings
                Set BreakLine to 0
```

4. Display / Print on screen the worst 5 underweight and the worst 5 obese patients in two groupings, male and female.

   To display the worst 5 underweight and 5 obese patients, we are going to use 2 temporary lists called Underweight and Obese. They will store all the patients with these 2 conditions. After the lists were created and all the data transferred into them, we will sort them descending after the BMI. We will keep the counting using a variable 'k' that will indicate us how many rows we have displayed on the screen.

```
Display_Worst_Underweight_and_Obese(List)
        Create Underweight list
        Set countUnderweight = 0
        Create Obese list
        Set countObese = 0
        for each Patient in size_of_the_List:
                if(Patient.Classification == 'Underweight'):
                        Set countUnderweight = countUnderweight + 1
                          Add Patient to Underweight list
                 if(Patient.Classification == 'Obese'):
                          Set countObese = countObese + 1
                        Add Patient to Obese list

        Check if countUnderweight = 0
                Display No records for underweight people
                else
                        Sort Underweight, descending after  BMI
                        Set k = 0
                        Display Female with underweight conditions:
                        for each Patient in size_of_the_Underweight:
                                if k == 5
                                    break from the loop
                                if Patient.Sex == 'F'
                                Display Patient_details
                                  Increment k by 1

                        Set k = 0
                        Display Male with underweight conditions:
                        for each Patient in size_of_the_Underweight:
                                if k == 5
                                    break from the loop
```

```
                        if Patient.Sex == 'M'
                        Display Patient_details
                        Increment k by 1


        Check if countObese = 0
                  Display No records for obese people
                else

                        Sort Obese, descending after  BMI
                        Set k = 0
                        Display Female with obese conditions:
                        for each Patient in size_of_the_Obese:
                                if k == 5
                                    break from the loop
                                if Patient.Sex == 'F'
                                Display Patient_details
                                  Increment k by 1

                        Set k = 0
                        Display Male with obese conditions:
                        for each Patient in size_of_the_Obese:
                                if k == 5
                                    break from the loop
                                if Patient.Sex == 'M'
                                Display Patient_details
                                Increment k by 1
```

5. Establish which patients need to be referred to a dietitian.
   The patients who need to be referred to a dietitian are the patients that have more than 2 conditions, are over 55 or they have a combination of these 2 conditions: asthmatic OR a smoker, Obese & suffers from hypertension.

6. Rank the order of priority according to the rules given above.
   To Rank the order of priority according to the given rules, we are going to check the number of conditions, for which conditions a patient suffers or if the patient is over 55, using a method called `RankOrder`:

```
RankOrder(List):
        Copy the elements from the List into a Temporary_List
for each Patient in List:
        create_a_new_attritute_for_Patient_class('Order')
        if Patient.Asthmatic = 'Y' or Patient.Smoker = 'Y' and Patient.Age>55
                Patient.Order = 1

 else if Patient.Classification = 'Obese' and Patient.Hypertension = 'Y'
                Patient.Order = 1

for each Patient in Temporary_List:
        create_a_new_attritute_for_Patient_class('NoConditions')
        if(Patient.Smoker == 'Y'):
                Patient.NoConditions = 1
        if(Patient.Asthmatic == 'Y'):
                Patient.NoConditions = Patient.NoConditions + 1
        if(Patient.NJT_NGR == 'Y'):
                Patient.NoConditions = Patient.NoConditions + 1
        if(Temporary_List[i].Hypertension == 'Y'):
                Patient.NoConditions = Patient.NoConditions + 1
        if(Temporary_List[i].RenalRT == 'Y'):
                Patient.NoConditions = Patient.NoConditions + 1
        if(Temporary_List[i].IleostomyColostomy == 'Y'):
                Patient.NoConditions = Patient.NoConditions + 1
```

```
        if(Temporary_List[i].ParenteralNutrition == 'Y'):
               Patient.NoConditions = Patient.NoConditions + 1

for each Patient in Temporary_List:
          if(Patient.NoConditions > 2):
              Patient.Order = 1#Top Priority
          if(Patient.NoConditions == 0):
                 Patient.Order = 4 #Less Priority
          if(Patient.NoConditions == 2):
                 Patient.Order = 2
          if(Patient.NoConditions > 0 and Patient.NoConditions < 2):
                 Patient.Order = 3
```

7. Display / Print on Screen patient names of those that need to be referred to a dietitian. Insert breaks every 10 patients to allow the user of the system to study the results.

   Like in the previous example with the 5 worst obese and 5 worst underweight, we are going to use the 'BreakLine' as an "indicator" for the number of rows displayed. To print on the screen the name of the patients that need to be referred to a dietitian we are going to use the last defined method , called `Referring_To_Dietitian`.

```
Referring_To_Dietitian(List):
for each Patient in size_of_the_List:
       Calculate the age for each patient based on their birthday
       create_a_new_attritute_for_Patient_class('age')
       Sort the List in descending order by age

Create a new temporary list called Temporary_List
for each Patient i in size_of_the_List:
              for each Patient j in size_of_the_List:
                     Check if Patient[i].Order = Patient[j].Order
                            Check if Patient[i].age > Patient[j].age
                                   Add Patient[i] into Temporary_List
                            else
                                   Add Patient[j] into Temporary_List

Delete the duplicates from Temporary_List
Sort the Temporary_List descending by the Order number of the patients
Set BreakLine = 0
for each Patient in size_of_the_Temporary_List:
       if Order attribute from Temporary_List is
       less than 4
             Display Patient.Name
       Set BreakLine = BreakLine + 1
       Check if BreakLine = 10
              Break a line after 10 displayings
              Set BreakLine = 0
```

At the end of the project, main method was created in which all the methods are called and display the desired output.