```python
#FUNCTION-1
from typing import List, Dict, Optional

def readPatientsFromFile(file_name):
    """
    Reads patient data from a plaintext file.

    fileName: The name of the file to read patient data from.
    Returns a dictionary of patient IDs, where each patient has a list
of visits.
    The dictionary has the following structure:
    {
        patientId (int): [
            [date (str), temperature (float), heart rate (int),
respiratory rate (int), systolic blood pressure (int), diastolic blood
pressure (int), oxygen saturation (int)],
            [date (str), temperature (float), heart rate (int),
respiratory rate (int), systolic blood pressure (int), diastolic blood
pressure (int), oxygen saturation (int)],
            ...
        ],
        patientId (int): [
            [date (str), temperature (float), heart rate (int),
respiratory rate (int), systolic blood pressure (int), diastolic blood
pressure (int), oxygen saturation (int)],
            ...
        ],
        ...
    }
    """
    patients= {}
#initialized an empty dictionary to store patient's data

    try:
#using try and except block to handle the errors which might come
during reading the file
       # Attempting to open the file
        with open(file_name, 'r') as file:
            for line_number, line in enumerate(file, start=1):
#using enumerate to get both line and its line number, starting from 1
                line = line.strip()
                fields = line.split(',')

                # Checking if the line has the correct number of
fields
                if len(fields) != 8:
                    print(f"Invalid number of fields ({len(fields)})
in line: {line_number}")
                    continue
```

```
                try:
                  #converting data to appropriate data types
                    patient_id = int(fields[0])
                    date = fields[1]
                    temperature = float(fields[2])
                    heart_rate = int(fields[3])
                    respiratory_rate = int(fields[4])
                    systolic_bp = int(fields[5])
                    diastolic_bp = int(fields[6])
                    oxygen_saturation = int(fields[7])

                    # Performing data verification checking
                    if temperature < 35 or temperature > 42:
                        print(f"Invalid temperature value
({temperature}) in line: {line_number}")
                        continue
                    if heart_rate < 30 or heart_rate > 180:
                        print(f"Invalid heart rate value
({heart_rate}) in line: {line_number}")
                        continue
                    if respiratory_rate < 5 or respiratory_rate > 40:
                        print(f"Invalid respiratory rate value
({respiratory_rate}) in line: {line_number}")
                        continue
                    if systolic_bp < 70 or systolic_bp > 200:
                        print(f"Invalid systolic blood pressure value
({systolic_bp}) in line: {line_number}")
                        continue
                    if diastolic_bp < 40 or diastolic_bp > 120:
                        print(f"Invalid diastolic blood pressure value
({diastolic_bp}) in line: {line_number}")
                        continue
                    if oxygen_saturation < 70 or oxygen_saturation >
100:
                        print(f"Invalid oxygen saturation value
({oxygen_saturation}) in line: {line_number}")
                        continue

                    # Adding the visit data to the patient's list of
visits
                    visit_data = [date, temperature, heart_rate,
respiratory_rate, systolic_bp, diastolic_bp, oxygen_saturation] #list
containing valid data values
                    if patient_id not in patients:
#checks if the id is in patients dict
                        patients[patient_id] = []
                    patients[patient_id].append(visit_data)

                except ValueError:
                    print(f"Invalid data type in line: {line_number}")
```

```
                    continue

    except FileNotFoundError:
        print(f"The file could not be found.")                    #if
the file not found then it generates an error
    except Exception as e:
        print("An unexpected error occurred while reading the file.")
        print(e)
#if any other error occurs then it gets caught by except block and
gets printed

    return patients

#FUNCTION-2

def displayPatientData(patients, patientId=0):
    """
    Displays patient data for a given patient ID.

    patients: A dictionary of patient dictionaries, where each patient
has a list of visits.
    patientId: The ID of the patient to display data for. If 0, data
for all patients will be displayed.
    """

    if patientId == 0:                                            #it
checks if the patientId is 0
        # displaying data for all patients
        for patient_id, visits in patients.items():
#iterating through each key, value pair in patients
            print(f"Patient ID: {patient_id}")
            for visit in visits:
#for each visit in the list of visits, it prints the data
                print(" Visit Date:", visit[0])
                print("  Temperature:", "%.2f" % visit[1], "C")
                print("  Heart Rate:", visit[2], "bpm")
                print("  Respiratory Rate:", visit[3], "bpm")
                print("  Systolic Blood Pressure:", visit[4], "mmHg")
                print("  Diastolic Blood Pressure:", visit[5], "mmHg")
                print("  Oxygen Saturation:", visit[6], "%")
            print()
#prints an empty line for separating the data for diff patients
    else:
        # displaying data for a specific patient ID
        if patientId in patients:
            print(f"Patient ID: {patientId}")
            visits = patients[patientId]
            for visit in visits:
                print(" Visit Date:", visit[0])
                print("  Temperature:", "%.2f" % visit[1], "C")
```

```
                print("  Heart Rate:", visit[2], "bpm")
                print("  Respiratory Rate:", visit[3], "bpm")
                print("  Systolic Blood Pressure:", visit[4], "mmHg")
                print("  Diastolic Blood Pressure:", visit[5], "mmHg")
                print("  Oxygen Saturation:", visit[6], "%")
        else:
            print(f"Patient with ID {patientId} not found.")

#FUNCTION-3

def displayStats(patients, patientId=0):
    """
    Prints the average of each vital sign for all patients or for the
specified patient.

    patients: A dictionary of patient IDs, where each patient has a
list of visits.
    patientId: The ID of the patient to display vital signs for. If 0,
vital signs will be displayed for all patients.
    """

    if type(patients) != dict:                                    #it
checks if the patients is not of type dict
        print("Error: 'patients' should be a dictionary.")
        return

    if type(patientId) != int:
#it checks if the 'patientId' parametere is not of type int
        print("Error: 'patient_id' should be an integer.")
        return

    if patientId == 0:
#if the patientId is 0 , then it is executed and display the stats for
all patients
        num_patients = len(patients)                              #it
counts the no. of patients by calculating the patient dicts length
        if num_patients == 0:
#checks if there are no patients
            print("No data found.")
            return

        temp_sum = hr_sum = rr_sum = sbp_sum = dbp_sum = spo2_sum =
num_visits = 0      #initialising sum of various signs=0 and
calculating the total number of visits across all patients

        for visits in patients.values():
#for all visits in patient dict
            for visit in visits:
#for each visit in the list of visits
                temp_sum += visit[1]
```

```
                hr_sum += visit[2]
                rr_sum += visit[3]
                sbp_sum += visit[4]
                dbp_sum += visit[5]
                spo2_sum += visit[6]
                num_visits += 1

        print("Vital Signs for All Patients:")
#printing the avg of signs for all patients
        print(" Average temperature:", "%.2f" % (temp_sum /
num_visits), "C")
        print(" Average heart rate:", "%.2f" % (hr_sum / num_visits),
"bpm")
        print(" Average respiratory rate:", "%.2f" % (rr_sum /
num_visits), "bpm")
        print(" Average systolic blood pressure:", "%.2f" % (sbp_sum /
num_visits), "mmHg")
        print(" Average diastolic blood pressure:", "%.2f" % (dbp_sum
/ num_visits), "mmHg")
        print(" Average oxygen saturation:", "%.2f" % (spo2_sum /
num_visits), "%")

    elif patientId > 0:                                  #if the
patientId provided will be greater than 0 then the stats will be
displayed for that particular patient only
        if patientId in patients:                            #if
patientId exists in the dictionary
            visits = patients[patientId]
#it takes the list of visits for the specific patient
            num_visits = len(visits)
#counting the length of the visits for that patient
            if num_visits == 0:
                print("No data found for patient with ID
{}.".format(patientId))
                return

            temp_sum = hr_sum = rr_sum = sbp_sum = dbp_sum = spo2_sum
= 0    #initialising sum of various signs=0 and calculating the total
number of visits across that specific patients

            for visit in visits:
                temp_sum += visit[1]
                hr_sum += visit[2]
                rr_sum += visit[3]
                sbp_sum += visit[4]
                dbp_sum += visit[5]
                spo2_sum += visit[6]

            print("Vital Signs for Patient {}:".format(patientId))
#printing the avg of signs for that specific patient
```

```
            print("  Average temperature:", "%.2f" % (temp_sum /
num_visits), "C")
            print("  Average heart rate:", "%.2f" % (hr_sum /
num_visits), "bpm")
            print("  Average respiratory rate:", "%.2f" % (rr_sum /
num_visits), "bpm")
            print("  Average systolic blood pressure:", "%.2f" %
(sbp_sum / num_visits), "mmHg")
            print("  Average diastolic blood pressure:", "%.2f" %
(dbp_sum / num_visits), "mmHg")
            print("  Average oxygen saturation:", "%.2f" % (spo2_sum /
num_visits), "%")
        else:
            print("No data found for patient with ID
{}.".format(patientId))                  #if the id not found then it
prints no data found for that patient

#FUNCTION 4

def addPatientData(patients, patientId, date, temp, hr, rr, sbp, dbp,
spo2, file_name):
    """
    Adds new patient data to the patient list.

    patients: The dictionary of patient IDs, where each patient has a
list of visits, to add data to.
    patientId: The ID of the patient to add data for.
    date: The date of the patient visit in the format 'yyyy-mm-dd'.
    temp: The patient's body temperature.
    hr: The patient's heart rate.
    rr: The patient's respiratory rate.
    sbp: The patient's systolic blood pressure.
    dbp: The patient's diastolic blood pressure.
    spo2: The patient's oxygen saturation level.
    fileName: The name of the file to append new data to.
    """
    try:
#if error will occur so we are using the try-except blocks
        year, month, day = map(int, date.split('-'))
#it splits the input date using a hyphen and then converts the strings
into intergers
        if not (1900 <= year <= 9999 and 1 <= month <= 12 and 1 <= day
<= 31):                  #checking conditions
            print("Invalid date. Please enter a valid date.")
            return
    except ValueError:
#if there occcurs an error while converting the components into
integers then this block is executed
        print("Invalid date format. Please enter date in the format
'yyyy-mm-dd'.")
```

```
        return
#checking whether the values are within the range or not
    if not (35.0 <= temp <= 42.0):
        print("Invalid temperature. Please enter a temperature between
35.0 and 42.0 Celsius.")
        return

    if not (30 <= hr <= 180):
        print("Invalid heart rate. Please enter a heart rate between
30 and 180 bpm.")
        return

    if not (5 <= rr <= 40):
        print("Invalid respiratory rate. Please enter a respiratory
rate between 5 and 40 bpm.")
        return

    if not (70 <= sbp <= 200):
        print("Invalid systolic blood pressure. Please enter a
systolic blood pressure between 70 and 200 mmHg.")
        return

    if not (40 <= dbp <= 120):
        print("Invalid diastolic blood pressure. Please enter a
diastolic blood pressure between 40 and 120 mmHg.")
        return

    if not (70 <= spo2 <= 100):
        print("Invalid oxygen saturation. Please enter an oxygen
saturation between 70 and 100%.")
        return

    if patientId not in patients:                       #if patientId
is not present in patients dict
        patients[patientId] = []
    patients[patientId].append([date, temp, hr, rr, sbp, dbp, spo2])
#appending the list containing those data into the list of visits for
that patient
    with open(file_name, 'a') as file:
        visit_str = ','.join(str(item) for item in [patientId, date,
temp, hr, rr, sbp, dbp, spo2])
        file.write(f"\n{visit_str}")
#it writes visit data to the file, creates a comma separated string of
the data and appends it to file
    print("Visit is saved successfully for Patient #", patientId)

#FUNCTION-5

def findVisitsByDate(patients, year=None, month=None):
    """
```

```
    Find visits by year, month, or both.

    patients: A dictionary of patient IDs, where each patient has a
list of visits.
    year: The year to filter by.
    month: The month to filter by.
    return: A list of tuples containing patient ID and visit that
match the filter.
    """

    filtered_visits = []
#initialising an empty list

    for patient_id, visits in patients.items():
#iterating through the items in patients dict
        for visit in visits:                                    #for each
visit in the list of visits
            date=visit[0]
#date is the first item in the visit list
            visit_year, visit_month, _ = date.split('-')
#splitting date into year and month by hyphen and using underscore for
day

            if (year is None or year==int(visit_year)) and (month is
None or month==int(visit_month)):         # if both of the conditions
met
                filtered_visits.append((patient_id, visit))
#then it gets appended to the visits

    return filtered_visits

#FUNCTION-6

def findPatientsWhoNeedFollowUp(patients):
    """
    Find patients who need follow-up visits based on abnormal vital
signs.

    patients: A dictionary of patient IDs, where each patient has a
list of visits.
    return: A list of patient IDs that need follow-up visits to to
abnormal health stats.
    """

    followup_patients = []                              #initialising an
empty list

    for patientId, visits in patients.items():          #iterating
through the items in patients dict
        for visit in visits:
```

```
            _, _,heart_rate, _, systolic_bp, diastolic_bp, spo2 =
visit                #unpacking the values from the visit list and
discarding the other values using underscores

            if heart_rate > 100 or heart_rate < 60 or systolic_bp >
140 or diastolic_bp > 90 or spo2 < 90:          #if any of these
conditions match then that patient needs followup
                followup_patients.append(patientId)
#and patient id is added to the followup list
                break
#exiting the inner loop

    return followup_patients

#FUNCTION-7

def deleteAllVisitsOfPatient(patients, patientId, file_name):
    """
    Delete all visits of a particular patient.

    patients: The dictionary of patient IDs, where each patient has a
list of visits, to delete data from.
    patientId: The ID of the patient to delete data for.
    filename: The name of the file to save the updated patient data.
    return: None
    """
    if patientId in patients:                          #it checks if
that id is in patients dictionary
        del patients[patientId]                                  #deletes
the patient's data of that particular person of that id
        with open(file_name, 'w') as file:
#opening a file in write mode using with statement to ensure that the
file is properly closedd after writing
            for patient_id, visits in patients.items():
#entering to the loop
                for visit in visits:
                    visit_str = ','.join(str(item) for item in visit)
#creates a comma separated string by joining each item in visit list
after converting them to strings
                    file.write(f"{patient_id},{visit_str}\n")
#writing the patient_id and the comma separated data to file
        print(f"Data for patient {patientId} has been deleted.")
    else:
        print(f"No data found for patient with ID {patientId}.")
#if id not found then it prints the statement

# FUNCTION-8: Analytics & Reporting

def analyticsReport(patients):
    """
```

```
    Generates analytics for all patients:
    1. Average vitals per patient
    2. List of high-risk patients
    3. Monthly visit trends
    """
    if not patients:  # Check if there is no data
        print("No patient data available.")
        return

    # 1. Average vitals per patient

    # This shows the overall health trend of each patient instead of
looking at one visit.

    print("\nAverage Vital Signs Per Patient:")
    for patient_id, visits in patients.items():  # Loop through each
patient
        num_visits = len(visits)  # Count visits for that patient
        # Calculate averages by summing each vital and dividing by
visit count
        avg_temp = sum(v[1] for v in visits) / num_visits
        avg_hr = sum(v[2] for v in visits) / num_visits
        avg_rr = sum(v[3] for v in visits) / num_visits
        avg_sbp = sum(v[4] for v in visits) / num_visits
        avg_dbp = sum(v[5] for v in visits) / num_visits
        avg_spo2 = sum(v[6] for v in visits) / num_visits
        # Print averages for this patient
        print(f"Patient {patient_id}: Temp={avg_temp:.2f}°C,
HR={avg_hr:.2f} bpm, RR={avg_rr:.2f}, SBP={avg_sbp:.2f},
DBP={avg_dbp:.2f}, SpO2={avg_spo2:.2f}%")

    # 2. High-risk patients

    # This flags people who might be in danger because their numbers
are not normal.

    followup_patients = findPatientsWhoNeedFollowUp(patients)  #
Reusing existing function
    if followup_patients:
        print("\nHigh-Risk Patients (Need Follow-up):")
        for patient_id in followup_patients:  # Print each risky
patient
            print(f"Patient {patient_id}")
    else:
        print("\nNo patients need follow-up.")

    # 3. Monthly visit trends

    # This shows when patients are coming the most.
```

```
    print("\nMonthly Visit Trends:")
    month_count = {}  # Dictionary to store visit counts per month
    for visits in patients.values():  # Loop through all patients
        for visit in visits:  # Each visit has date + vitals
            year, month, _ = visit[0].split('-')  # Extracting year
and month from date
            key = f"{year}-{month}"  # Example: "2022-08"
            month_count[key] = month_count.get(key, 0) + 1  # Count
visits per month
    for key in sorted(month_count.keys()):  # Sort months in order
        print(f"{key}: {month_count[key]} visits")  # Print monthly
visit count

#MAIN FUNCTION


def main():
    patients= readPatientsFromFile('patients.txt')
#reads the patient's data from file using functn and stores the data
in patient dict
    while True:
#displaying opts to user
        print("\n\nWelcome to the Health Information System\n\n")
        print("1. Display all patient data")
        print("2. Display patient data by ID")
        print("3. Add patient data")
        print("4. Display patient statistics")
        print("5. Find visits by year, month, or both")
        print("6. Find patients who need follow-up")
        print("7. Delete all visits of a particular patient")
        print("9. Analytics & Reporting")
        print("8. Quit\n")

        choice = input("Enter your choice (1-8): ")
#asking for the user's choice
        if choice == '1':
#displaying every patient's data
            displayPatientData(patients)
        elif choice == '2':
#displaying data for a specific patient id
            patientID = int(input("Enter patient ID: "))
            displayPatientData(patients, patientID)
        elif choice == '3':
#adds new medical visit for a patient
            patientID = int(input("Enter patient ID: "))
            date = input("Enter date (YYYY-MM-DD): ")
            try:
#it checks the conditions
                temp = float(input("Enter temperature (Celsius): "))
                hr = int(input("Enter heart rate (bpm): "))
```

```
                rr = int(input("Enter respiratory rate (breaths per
minute): "))
                sbp = int(input("Enter systolic blood pressure (mmHg):
"))
                dbp = int(input("Enter diastolic blood pressure
(mmHg): "))
                spo2 = int(input("Enter oxygen saturation (%): "))
                addPatientData(patients, patientID, date, temp, hr,
rr, sbp, dbp, spo2, 'patients.txt')
#updates the file with new patient data
            except ValueError:
                print("Invalid input. Please enter valid data.")
        elif choice == '4':
#it calculates and displays the avg vital signs for either all
patients or for a specific patient
            patientID = int(input("Enter patient ID (or '0' for all
patients): "))
            displayStats(patients, patientID)
        elif choice == '5':
#it filters visits based on the given month or year
            year = input("Enter year (YYYY) (or 0 for all years): ")
            month = input("Enter month (MM) (or 0 for all months): ")
            visits = findVisitsByDate(patients, int(year) if year !=
'0' else None,
                                      int(month) if month != '0' else
None)
            if visits:
                for visit in visits:
                    print("Patient ID:", visit[0])
                    print(" Visit Date:", visit[1][0])
                    print("  Temperature:", "%.2f" % visit[1][1], "C")
                    print("  Heart Rate:", visit[1][2], "bpm")
                    print("  Respiratory Rate:", visit[1][3], "bpm")
                    print("  Systolic Blood Pressure:", visit[1][4],
"mmHg")
                    print("  Diastolic Blood Pressure:", visit[1][5],
"mmHg")
                    print("  Oxygen Saturation:", visit[1][6], "%")
            else:
                print("No visits found for the specified year/month.")
        elif choice == '6':
#checks each patients visits for specific conditions and compiles a
list of patientId's who need followup
            followup_patients = findPatientsWhoNeedFollowUp(patients)
            if followup_patients:
                print("Patients who need follow-up visits:")
                for patientId in followup_patients:
                    print(patientId)
            else:
```

```
                print("No patients found who need follow-up visits.")
        elif choice == '7':
#it removes all visits of a specific patient and updates the file
            patientId = input("Enter patient ID: ")
            deleteAllVisitsOfPatient(patients, int(patientId),
"patients.txt")
        elif choice == '9':
            analyticsReport(patients)
        elif choice == '8':
#exiting
            print("Goodbye!")
            break
        else:
#if any other choice is given other than 1-8 then it prints invalid
choice
            print("Invalid choice. Please try again.\n")
if __name__ == "__main__":
    main()
```

```
Welcome to the Health Information System


1. Display all patient data
2. Display patient data by ID
3. Add patient data
4. Display patient statistics
5. Find visits by year, month, or both
6. Find patients who need follow-up
7. Delete all visits of a particular patient
9. Analytics & Reporting
8. Quit

Enter your choice (1-8): 9

Average Vital Signs Per Patient:
Patient 1: Temp=37.12°C, HR=71.00 bpm, RR=17.60, SBP=120.40,
DBP=80.00, SpO2=94.60%
Patient 2: Temp=37.62°C, HR=73.00 bpm, RR=19.60, SBP=128.20,
DBP=84.00, SpO2=95.20%
Patient 3: Temp=36.94°C, HR=68.40 bpm, RR=16.20, SBP=118.40,
DBP=75.40, SpO2=93.40%
Patient 4: Temp=37.12°C, HR=71.60 bpm, RR=19.40, SBP=124.80,
DBP=82.40, SpO2=96.80%
Patient 5: Temp=37.48°C, HR=71.60 bpm, RR=18.60, SBP=123.60,
DBP=81.60, SpO2=94.60%
Patient 6: Temp=37.06°C, HR=73.00 bpm, RR=19.00, SBP=124.00,
DBP=82.00, SpO2=98.00%
Patient 7: Temp=37.31°C, HR=70.57 bpm, RR=19.57, SBP=120.57,
```

```
DBP=80.57, SpO2=94.57%
Patient 8: Temp=37.64°C, HR=75.00 bpm, RR=22.00, SBP=130.00,
DBP=86.00, SpO2=97.00%
Patient 10: Temp=36.94°C, HR=70.00 bpm, RR=19.00, SBP=122.00,
DBP=80.00, SpO2=94.00%

No patients need follow-up.

Monthly Visit Trends:
2020-08: 3 visits
2021-02: 3 visits
2021-08: 3 visits
2022-02: 3 visits
2022-05: 1 visits
2022-06: 1 visits
2022-07: 1 visits
2022-08: 4 visits
2022-09: 1 visits
2022-10: 1 visits
2022-11: 1 visits
2022-12: 1 visits
2023-01: 1 visits
2023-02: 1 visits
2023-03: 2 visits
2023-04: 1 visits
2023-05: 1 visits
2023-06: 1 visits
2023-07: 1 visits
2023-08: 1 visits
2023-09: 2 visits
2023-10: 1 visits
2023-11: 1 visits
2023-12: 1 visits
2024-01: 1 visits
2024-02: 1 visits
2024-03: 1 visits
2024-04: 1 visits
2024-05: 1 visits
2024-06: 1 visits
2024-07: 1 visits
2024-08: 1 visits
2024-09: 1 visits
2024-10: 1 visits


Welcome to the Health Information System


1. Display all patient data
2. Display patient data by ID
```

```
3. Add patient data
4. Display patient statistics
5. Find visits by year, month, or both
6. Find patients who need follow-up
7. Delete all visits of a particular patient
9. Analytics & Reporting
8. Quit

Enter your choice (1-8): 8
Goodbye!
```