MCD4710 - Introduction to Algorithms and Programming
Assignment 1 (8%)

Due: Thursday, July 28, 2022, 11:55 pm - Week 6

Objectives

The objectives of this assignment are:

- To gain experience in designing algorithms for a given problem description and implementing those algorithms in Python.
- To demonstrate your understanding on:
    - Implementing problem solving strategies
    - Recognizing the relationship between a problem description and program design
    - Decomposing code into functions in Python.

Submission Procedure

Your assignment will not be accepted unless it meets these requirements:

1. Your name and student ID must be included at the start of every file in your submission.
2. Save your file(s) into a zip file called YourStudentID.zip
3. Submit your zip file containing your entire submission to Moodle.

Late Submission

Late submission will have 5% deduction of the total assignment marks per day (including weekends). Assignments submitted 7 days after the due date will not be accepted.

Important Notes

- Please ensure that you have read and understood the university's procedure on plagiarism and collusion available at https://www.monashcollege.edu.au/__data/assets/pdf_file/0005/1266845/Student-Academic-Integrity-Diplomas-Procedure.pdf . You will be required to agree to these policies when you submit your assignment.
- Your code will be checked against other students' work and code available online by advanced plagiarism detection systems. Make sure your work is your own. Do not take the risk.
- Your program will be checked against a number of test cases. Do not forget to include comments in your code explaining your algorithm. If your implementation has bugs, you may still get some marks based on how close your algorithm is to the correct algorithm. This is made difficult if code is poorly documented.

Assignment code interview

Each student will be interviewed during a lab session regarding their submission to gauge your personal understanding of your Assignment code. The purpose of this is to ensure that you have completed the code yourself and that you understand the code submitted. Your assignment mark will be scaled according to the responses provided.

Interview Rubric

0	The student cannot answer even the simplest of questions. There is no sign of preparation. They probably have not seen the code before.
0.25	There is some evidence the student has seen the code. The answer to at least one question contained some correct points. However, it is clear they cannot engage in a knowledgeable discussion about the code.
0.5	The student seems underprepared. Answers are long-winded and only partly correct. They seem to be trying to work out the code as they read it. They seem to be trying to remember something they were told but now can’t remember. However, they clearly know something about the code. With prompting, they fail to improve on a partial or incorrect answer.
0.75	The student seems reasonably well prepared. Questions are answered correctly for the most part but the speed and/or confidence they are answered with is not 100%. With prompting, they can add a partially correct answer or correct an incorrect answer.
1	The student is fully prepared. All questions were answered quickly and confidently. It is absolutely clear that the student has performed all of the coding themselves.

Marking Criteria

This assignment contributes to 8% of your final mark.

Task 1 – Read data: 3 marks

Task 2 – Convert weight to kg: 2 marks

Task 3 – Calculate BMI: 2 marks

Task 4 – Categorise patients: 3 marks

Task 5 – Get BMI: 2 marks

Task 6 – Get blood pressure: 2 marks

Task 7 – Check smoking status: 2 marks

Task 8 – Extract high-risk records: 4 marks

Programming practice – 2 marks

- Decomposition (1 mark)
- Variable names and code Documentation (1 mark)

Total: 22 marks

MCD4710 - Introduction to Algorithms and Programming
Assignment 1 (8%)

Background Information

A data set containing patient information from a hospital (a modified version of a data set taken from MathWorks sample data sets: *hospital.mat*) is provided in a CSV file (*hospital_data.csv*) and contains records of 100 patients with the 8 fields; name, gender, age, weight (lbs), height (cm), smoking status, systolic blood pressure and diastolic blood pressure.

You are required to write functions to achieve the following (note that this dataset is just for the purposes of testing, and your module must work for any similar dataset as well).

Task 1: Read the data into a table

Implement a function named `read_data(filename)`, with the following specification, which reads a given file and returns the contents in a table. Make sure you store the smoking status (in the 6th column) as an integer and the other numerical values as floats.

Hint: you can use the `isalpha(string)` string method (you may assume none of the fields in the data file are alphanumeric).

Input: The name of the data file (`filename`)
Output: A table with data in the given file.

For example:
```
>>> patient_data = read_data(‘hospital_data.csv’)
>>> patient_data[:3] # The first three rows of the table
[['SMITH', 'M', 38.0, 176.0, 145.0, 1, 124.0, 93.0], ['JOHNSON', 'M', 43.0, 163.0,
146.0, 0, 109.0, 77.0], ['WILLIAMS', 'F', 38.0, 131.0, 147.0, 0, 125.0, 83.0]]
```

Task 2: Convert the weight to kilograms

The weight of each patient is given in pounds (lbs). Write a function named `lbs_to_kg(table)`, which converts the weight to kilograms (kg) and replaces the original weight in the table, for each patient. Make sure the weight in kg is rounded to one decimal place.
Note: 1 lbs = 0.4536 kg

Input: A table containing all the patient data that is available in the given data file.
Output: The table with weight converted to kilograms

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> lbs_to_kg(patient_data)
>>> patient_data[:3] # The first three rows of the modified table
[['SMITH', 'M', 38.0, 79.8, 145.0, 1, 124.0, 93.0], ['JOHNSON', 'M', 43.0, 73.9,
146.0, 0, 109.0, 77.0], ['WILLIAMS', 'F', 38.0, 59.4, 147.0, 0, 125.0, 83.0]]
```

Task 3: Calculate the body mass index (BMI)
The body mass index (BMI) for each patient needs to be calculated. Write a function named `calculate_BMI(table)`, which calculates the BMI for each person, using the formula below. Append the BMI values, rounded to one decimal place, as the last column of the `table`.

$$\text{BMI}=\frac{\text{mass (kg)}}{\text{height}^2\ (\text{m}^2)}$$

Input: A table containing all the patient data that is available in the given data file, with the weight converted to kg.
Output: The table with the BMI values added to the last column of the table.

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> lbs_to_kg(patient_data)
>>> calculate_BMI(patient_data)
>>> patient_data[:3] # The first three rows of the modified table
[['SMITH', 'M', 38.0, 79.8, 145.0, 1, 124.0, 93.0, 38.0], ['JOHNSON', 'M', 43.0,
73.9, 146.0, 0, 109.0, 77.0, 34.7], ['WILLIAMS', 'F', 38.0, 59.4, 147.0, 0, 125.0,
83.0, 27.5]]
```

Task 4: Categorize based on BMI
The patients in the data table need to be categorized according to their BMI value. Write a function named `categorise(table)` that determines the category for each person based on the following classification and store characters U, N or O for underweight, normal and obese, respectively. This information should be stored next to their BMI value in the table. The categorization should be done according to the following classification.

$$BMI < 18 : Underweight$$
$$18 \leq BMI < 30 : Normal$$
$$BMI \geq 30 : Obese$$

Input: A table containing all the patient data that is available in the given data file, with the BMI values appended.
Output: The table with the categories added to the last column of the table.

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> lbs_to_kg(patient_data)
>>> calculate_BMI(patient_data)
>>> categorise(patient_data)
>>> patient_data[:3] # The first three rows of the modified table
[['SMITH', 'M', 38.0, 79.8, 145.0, 1, 124.0, 93.0, 38.0, ‘O’], ['JOHNSON', 'M',
43.0, 73.9, 146.0, 0, 109.0, 77.0, 34.7, ‘O’], ['WILLIAMS', 'F', 38.0, 59.4,
147.0, 0, 125.0, 83.0, 27.5, ‘N’]]
```

Task 5: Look up BMI value of a given patient

Write a function named `get_BMI(table,name)` that returns the BMI value of a given patient by looking up the name in the `table`. If the given name is not available in the table, the function should return `None`. You may assume that the names are unique.

Input: A table containing all the patient data that is available in the given data file, with the BMI values appended and the name of a patient.
Output: The BMI value of the given patient.

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> lbs_to_kg(patient_data)
>>> calculate_BMI(patient_data)
>>> get_BMI(patient_data,’Johnson’)
34.7
```

Task 6: Look up blood pressure value of a given patient

Write a function named `get_BP(table,name)` that returns the blood pressure readings as a tuple, (systolic, diastolic), of a given patient by looking it up in the `table`. Make sure the blood pressure readings are returned as integer values. If the given name is not available in the table, the function should return `None`. You may assume that the names are unique.

Input: A table containing all the patient data that is available in the given data file and the name of a patient.
Output: The blood pressure readings of the given patient as a tuple.

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> get_BP(patient_data,’Johnson’)
(109, 77)
```

Task 7: Determine whether a given patient is a smoker or not

Write a function named `is_smoker(table,name)` that returns `True` if the given patient is a smoker and `False` if not. The function should return `None` if the patient record is not available in the table.

Input: A table containing all the patient data that is available in the given data file and the name of a patient.
Output: True if the patient is a smoker and False otherwise.

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> is_smoker(patient_data,’Johnson’)
False
>>> is_smoker(patient_data,’White’)
True
```

Task 8: Extract the records of high-risk patients and write to a separate csv file

Now we need to extract the records of high-risk patients from the table and save their details (all column values of the patient in the table) into a new csv file. Write a function named `extract_high_risk(table,filename)` that extracts the records of patients who satisfy **all** of the following conditions and write their data into the given file, `output_filename`. Add the column headings to the file as well.
A patient is deemed at high-risk if they

- are obese
- has either systolic blood pressure greater than 125 or diastolic blood pressure greater than 85.
- are a smoker

Note: You have to use the functions written in the previous tasks. Do not rewrite the code written in the previous tasks here.

Input: A table containing all the patient data and a filename
Output: The given file updated with the high-risk patient details

For example:
```
>>> patient_data = read_data('hospital_data.csv')
>>> lbs_to_kg(patient_data)
>>> calculate_BMI(patient_data)
>>> categorise(patient_data)
>>> extract_high_risk(patient_data,'high_risk_patients.csv')
>>> high_risk_table = read_data('high_risk_patients.csv') # read high-risk values
(for testing)
```

```
>>> high_risk_table[:2] # The first two rows of the high-risk records
[['SMITH', 'M', 38.0, 79.8, 145.0, 1, 124.0, 93.0, 38.0, 'O'], ['WHITE', 'M',
39.0, 91.6, 152.0, 1, 130.0, 95.0, 39.6, 'O']]
```

---------------------------------- END OF ASSIGNMENT ---------------------------------------