Master of Computer Applications
MCAC-101: Object Oriented Programming
Unique Paper Code: 223401101
Semester I
March-2022
Year of admission: 2021

Time: Three Hours **Max. Marks: 70**

Notes:

1. Answer any **Four** questions. All questions carry equal marks.
2. You MUST document your code properly for full credit.

1. a. Happy Numbers: A number is called a happy number, if you repeat the process of the sum of the square of the digits, till the value 1 is obtained. The steps to check a number (`n`) is Happy Number are: (i) compute the sum of the squares of its digits (ii) if the resulting value is 1, then `n` is a happy number, else repeat step (i). If a number is not a happy number, there will be an endless loop/cycle to this execution.
   For example, 19 is a happy number since,
   STEP1: 1^2 + 9^2 = 82
   STEP2: 8^2 + 2^2 = 68
   STEP3: 6^2 + 8^2 = 100
   STEP4: 1^2 + 0^2 + 0^2 = 1

   Given this context, you are required to write two Python functions:
   - A recursive function `SSD` to find the sum of the square of the digits. The function will accept a number and returns the sum of the square of the digits.
   - A function `checkHappyNum` that accepts a number `n` and returns `True`, if `n` is a happy number and `False` if the `n` is an unhappy number. To avoid an endless loop/cycle, you can stop after 6 iterations.

   b. Write a Python function `mergeLists` that takes two lists as arguments and returns a dictionary of merged lists of elements in sorted order.
   Input: Two lists that may contain strings and numbers
   Output: a dictionary with two keys:
   `mergedStr`: a list of merged strings in English-dictionary order
   `mergedNum`: a list of merged numbers in ascending order.
   The task includes:
   - Separating numbers and strings from the inputted lists
   - Merging of numbers from the two lists and sorted them in ascending order

- Merging of the strings in English-Dictionary order

For example,

```
L1=["Morning", 100, "hey", "Hi", 23, 78, "Good",
"Evening", -9]
L2=[89, "Rain", "Sunny", "Cloudy", 91, 107, "Humid"]

mergeLists(L1, L2) returns a dictionary as

{"mergedStr": ["Cloudy", "Evening", "Good", "Hi",
"Humid", "Morning", "Rain", "Sunny", "hey"]
"mergedNum":[-9, 23, 78, 89, 91, 100,107]
}
```

You can define more user-defined functions, if required.

2. Write a modular Python program for spell checker, where you have a database of words stored in a text file named "`myWords.txt`".

   Input: Filename, containing text for spell checking and name is inputted by the user.

   Output: Return a list of words that are not present in `myWords.txt`. Such words may or may not have the correct spelling.

   Kindly note the following:

   - `myWords.txt` contains a list of unique words separated by a new line in English-dictionary order i.e., there exists only one word per line.
   - Ignore case while searching a word in `myWords.txt`, i.e., "`this`" and "`This`" are the same words.
   - For each word that is not found in the `myWords.txt` file, the program should prompt the user to add the word to `myWords.txt`. If a user opts to add the word to the file, then it should be added in such a way that the file should maintain English-dictionary order. A user may add any number of words to the `myWords.txt` file from the returned list.
   - A word may occur more than once in the inputted file.

   You can define any number of user-defined functions. Also, handle all possible exceptions that can be raised.

3. a. Write a Python function `myTelephoneDict` to find the phone number of a person. There exists a telephone directory that stores the phone number and the name of the person. The function accepts the name of the person and returns the phone number of the person if it exists in the dictionary otherwise returns "No Record found". Choose the function parameters appropriately.

   Kindly note the following:

   - The telephone directory is maintained using dictionary data type where the phone number is used as a key and the corresponding value holds the name of the person.

- If there exists more than one key-value pair for a given name, return all the phone numbers.

b. Create Python functions for performing the following operations:
   - Given a list of elements, for each element of the list, check whether it is a palindrome or not. The function should return a list of True/False, Ture represents that the respective element is a palindrome, and False means the element is not a palindrome.
   - Function takes two strings as an input parameter and prints all the interleaving of the given two strings preserving their order of occurrence. For example, interleaving of strings AB and CD should be ABCD, ACBD, ACDB, CDAB, CADB, and CABD.

   Choose the parameter(s) of Python functions appropriately.

4. a. Given a list of jobs, a schedular schedules the jobs. Kindly note the following:
   - A job is represented by a name, burst time (in time units, time to require a job to schedule), priority value (an integer value, the high value represents a low priority, and vice-versa), and arrival time (time at which job arrives for execution).
   - A job can arrive at any moment of the time. So, give provision to a user to add a new job.
   - Scheduling procedure: All jobs are stored and executed in the order of their arrival time if the priority of all jobs is the same. Otherwise, jobs with higher priority will be executed first based on the arrival time.

   Choose appropriate data type(s) to represent jobs and create a modular program in Python for the following.
   (i) To print the scheduled jobs in the order of their execution.
   (ii) To return waiting time for each job, where the waiting time is the difference between the CPU allotted time and arrival time.
   (iii) To enter a new job.
   (iv) To represent details of a particular job.

   Also, write a menu-driven driver code to carry out all operations.

5. Create a class `Matrix` class in Python that has the following private instance variable:
   `nRow`: number of rows
   `nCol`: number of columns
   `elem`: to hold elements of the matrix
   An object of the class `Matrix` represents a 2D-matrix.

   Also, write methods for each of the following:
   - To get and set the number of rows using a property decorator.
   - To get and set the number of columns using a property decorator.

- To get and set the elements of the matrix at a given position `(i,j)`.
- To perform Addition/subtraction of two matrices and return a `Matrix` object containing the result of the operation. If the operation is not possible, return "Operation can't be performed".
- Multiplication of the two matrices. If the matrices are not multipliable, "Matrices cannot be multiplied" will be displayed. Further, if a matrix is 1D i.e. either `nRow` or `nCol` is 1, then, perform dot product, provided the vectors are multipliable, otherwise display a message "Dot product is not possible".
- To delete an object of `Matrix` class. Also, keep a track of the number of matrix objects.
- To print an object of `Matrix` class.

Apply proper validation for creating an object of `Matrix` class and application of operations, wherever required.

6. Create Python functions to read the contents of a text file named **`vaccinationDetails.txt`** containing vaccination information of Delhi in the following format:

`Name, Age, Gender, District, firstVaccinationDate, secondVaccinationDate`

where `Name:` name of the person

`Age`: age of the person

`Gender:` Gender of the Person

`District`: District to which the person belongs

`firstVaccinationDate`: Date when the person received the first vaccination

`secondVaccinationDate:` Date when the person received a second vaccination

Write python functions for the following:

a. To display a bar chart showing the total number of records per district
b. To display district-wise distribution of the number of teenagers (age between 13-19 years) who received only one vaccination and both vaccinations
c. To display a pie chart showing the number of Males and Females vaccinated
d. To find the total number of persons who received both vaccinations district-wise

Label the axis appropriately and use legends, wherever required. Also, handle all possible exceptions that can be raised.

Kindly note the following:

- For example, the file `vaccinationDetails.txt` appears as follows:
  Sachin Kumar, 18, M, North Delhi, 10.11.2021, 02.01.2022
  Manisha, 23, F, Central Delhi, 22.11.2021, 12.02.2022
  Amir Khan, 45, M, North Delhi, 12.11.2021
  Zahir, 60, M, East Delhi, 10.01.2021, 23.03.2021
  Meera Rajput, 16, F, Central Delhi, 13.02.2022
  Sahr, 34, F, North Delhi,

Sahil Sharma, 22, M, South Delhi, 11.04.2021, 13.06.2021

- Each line represents a record.
- There are 11 districts in Delhi, namely Central Delhi, East Delhi, New Delhi, North Delhi, North East Delhi, North West Delhi, Shahdara, South Delhi, South East Delhi, South West Delhi, West Delhi.
- Gender may have values M (for male), F (for female), O (for others)
- Ignore the empty line(s) or a record with no vaccination information
- Represent zero in the graph, if no relevant record is found