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

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

Instructions:

1. All questions are compulsory.
2. Attempt all the parts of a question together
3. You **MUST** document your code properly for full credit.

1. a. Write a python function `heterogram` that accepts a string, and checks whether an entered string is a heterogram or not. The function returns `True` if the input string is a hetrogram otherwise, returns a set of the duplicate alphabets present in the string. 7

   Note: A word, phrase, or sentence is called a heterogram in which no letter of the alphabet occurs more than once.

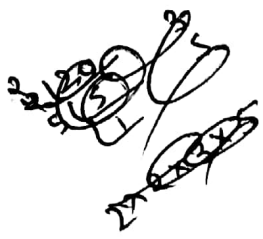
   b. Consider the following scenario: two players, say `P1` and `P2` are running around a circular field and took `x` and `y` minutes to complete one round respectively. If both started simultaneously and go in the same direction, the coach wants to compute the time after which they will meet at the starting point. 7

   Help the coach by writing a Python function `myMeetTimeCalc` that accepts the time (in minutes) taken by two players to complete one round of the field as parameters and returns the time after which they'll meet at the starting point.

   For example, if P1 and P2 took 20 minutes and 15 minutes respectively to complete one round of the field, the function `myMeetTimeCalc(20,15)` return `60mins`. LCM

2. Write a Python function `strAnalysis` to find the longest word(s), and smallest word(s) except articles, present in a file. The function accepts the name of a text file as input and returns a dictionary containing a list of longest, smallest, and unique words with keys `longWords`, `shortWords`, and `uniqueWords` respectively. Also, handle all possible exceptions that can be raised. 14

   Note:
   - The text file contains only strings.
   - The function should ignore articles to compute the longest or shortest word.



- Articles should be included while computing unique words.
- Unique words should be identified after ignoring the case.

For example, if the file named "data.txt" contains:

```
This is my First Python Program.
I am excited to learn Python.
A program is a set of instructions to solve a problem.
```

The function `strAnalysis("data.txt")` should return:

```
{"longWords": [instructions]
"shortWords": [I]
"uniqueWords": [This, is, my, First, Python, Program,
I, am, excited, to, learn, A, set, of, instructions,
solve, problem]
}
```

3. Write a recursive python function `dataManipulation` with two parameters to carry out the following: 14

- If input parameters are integers, then carry out the multiplication of two integer numbers **without** using the multiplication operator.
  For example, `dataManipulation(5,3)` returns `15`.
- If input parameters are strings, then concatenate the strings **without** using the concatenation operation ('+') or in-built function.
  For example, `dataManipulation("Python ","Program")` returns `"Python Program"`.
- If input parameters are string and integers say `str1`, and `n` respectively, then concatenate the string `n` number of times **without** using concatenation operation ('+') or an in-built function.
  For example, `dataManipulation("Python",3)` returns `"PythonPythonPython"`.
- Otherwise return "invalid inputs"

4. Create a class `Vector` in Python that has the following private instance variable: 14

`n`: number of components

`elems`: to hold elements of the vector

An object of the class `Vector` represents a coordinate of points in `n-dimensional` space.

Also, write methods for each of the following:

- To get and set the number of components using a property decorator.
- To get and set the elements of the vector using a property decorator.
- To translate a given `vector`. If the operation is not possible, return "Operation can't be performed".
- To delete an object of the `Vector` class. Also, keep a track of the number of points.

- To print an object of `Vector` class.

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

5 Write a modular Python program to carry-out merging of two lists. Define the following functions for the program: 14

- `checkSorted(slist)`: this function accepts a list and returns a tuple with two values: `(status, order)` where
  - `status` can have two values: `True` or `False`, where `True` represents that the given list is sorted and `False` otherwise.
  - `order` can have three values: `'asc'`, `'desc'` or `None`, where `'asc'` represents that the list is sorted in ascending order, `'desc'` for descending, and `None` for unsorted list.

  For example,
  `checkSorted([10, 20, 30])` returns `(True,'asc')`
  `checkSorted([40, 20, 10, 5])` returns `(True,'desc')`
  `checkSorted([10, 100, 20, -30])` returns `(False,None)`

- `convertAsc(slist)`: this function accepts a list and returns a list sorted in ascending order only.
  For example,
  `convertAsc([40, 20, 10, 5])` returns `[5, 10, 20, 40]`
  `convertAsc([10, 100, -10, 5])` returns `[-10, 5, 10, 100]`

- `myMergeList(slist1, slist2)`: this function accepts only sorted lists in ascending order and returns a third sorted list by merging these two input lists into one sorted list in a single scan of the lists.
  For example,
  `myMergeList([10, 20, 30], [5, 19, 130])` returns `[5, 10, 19, 20, 30, 130]`

Note: You can use in-built functions.