Unique Paper Code : 32343307
Name of Course : B.Sc. (H) Computer Science (LOCF)
Name of Paper : Programming in Python (SEC)
Semester : III
Year of Admission : 2019

Duration: 3 Hours **Max Marks : 75**

Attempt any **FOUR** questions.
All questions carry equal marks.
You must document your code properly for full credit.

Q1. Write a Python function `check_Eligibility(n)` which accepts an argument `n`. It displays a message `'Proceed for Printing'` and returns `'True'` in case `n` is a prime number less than `29`, otherwise it displays `'Sorry'` and returns `'False'`.

Write another Python function `displayPattern(n)` which accepts an argument `n`. It calls `check_Eligibility(n)`. The function `displayPattern(n)` displays the following pattern with `n` rows, only if the function `check_Eligibility(n)` returns `'True'`, otherwise it displays `'Pattern Not Possible'` and exits from the function. For `n=5`, the pattern will generate `5` rows as given below.

```
A
B * B
C *** C
D ***** D
E ******* E
```

Q2. Consider the following strings:

```
str1 = "Ronald Brown"
str2 = "Richard Brown"
str3 = "Harry/* Potter% is ^a fictional !! character-&"
```

- Write a Python function `append_Strings()` to create a new string, `str4` by appending `str1` and `str2` without using in-built Python functions.
- Write Python code snippet to arrange the characters of `str4` so that all lowercase letters should come first. Also, find all the occurrences of substring "`row`" in `str4` ignoring the case; without using in-built Python functions.
- Using an in-built Python function, write a statement to remove all the special symbols from `str3`.

Consider two lists `List1` and `List2` as shown below.

```
List1= [5,10,15,20,25,30,35]
List2= [10,20,30,40,50,60,70]
```

- Write Python code snippet to add a new element `40` after `35` in `List1`. The modified `List1` should be

```
List1 = [5,10,15,20,25,30,35,40]
```

- Write a Python function that appends the elements of `List1` into `List2` and returns the appended list as

```
List2 = [10,20,30,40,50,60,70,5,10,15,20,25,30,35,40]
```

- Write a Python function to remove all the duplicate values from the list `List2`.

Q3. Consider dictionary `Dict1` that represents vehicle models and its number available in a company. For example, `Dict1` is defined as follows:

```
Dict1 = {'BMW' :5, 'Mercedes': 10, 'Volkswagen': 10,
         'Jaguar': 4, 'Landrover':15}
```

- Write a Python function `Model()` that accepts `Dict1` along with a vehicle model. It returns the number of vehicles available for that model. If the model does not exist in `Dict1`, then it should return a value of `-1`.

- Write a Python function `Change()` to accept `Dict1` along with a vehicle model `'VM'`. The function should return the updated dictionary `Dict1` as follows:

    - Case 1: if `VM` exists in the dictionary and the number of models corresponding to `VM` is less than `5`, then `VM` should be removed from the dictionary.
    - Case2: if `VM` exists in the dictionary and the number of models corresponding to `VM` is greater than `5`, then decrease the number of models by 2.

Consider the following tuples:

```
Tuple1 = (11,22,33)
Tuple2 = (99,88,77)
```

- Write a Python function to swap the values of `Tuple1` and `Tuple2`. The expected output is as follows:

```
Tuple1 = (99,88,77)
Tuple2 = (11,22,33)
```

- Write a Python function `Div3and5()` that takes a `10`-element numeric tuple and returns the following:

    - sum of elements which are divisible by `3`.
    - sum of elements which are divisible by `5`.
    - sum of elements which are divisible by `3` and `5`.

Consider the following sets and give Python code snippets for each of the following:

```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}
```

- Display common elements in `set1` and `set2`.
- Display a new set of elements such that the elements are present in either `set1` or `set2` but not in both the sets.

Q4. Write a Python code snippet to read a text file, `'file1.txt'`, line by line. For each line read from `'file1.txt'`, split the text into words and count the number of characters and vowels in each word. Write each word along with its character count and vowel count separated by a question mark(`?`) in file `'file2.txt';` with each word and its corresponding character count and vowel count in separate lines. Display the contents of `'file2.txt'` on the console.

Handle the exception if `'file1.txt'` does not exist.

For example, the contents of `'file1.txt'` are as shown below. Python code snippet should create `'file2.txt'` having contents as shown below:

"file1.txt"	**"file2.txt"**
Hello World	Hello?5?2
Python is interesting	World?5?1
	Python?6?1
	Is?2?1
	Interesting?11?4

Q5. Write a Python program that accepts a list `Number_List` as input. Ensure that user enters numeric values as input in the list. Take an input `Num` from the user to be searched in `Number_List`.

- Define a Python function `SortAndSearch()` with two arguments as `Number_List` and `Num`. Use linear search in `SortAndSearch()` to search given number `Num` in `Number_List`. Also, `SortAndSearch()` displays the position of `Num` if it is found in `Number_List` otherwise displays the message `'Number is not found in the list'`.

- Use insertion sort in `SortAndSearch()` to arrange the elements of `Number_List` in ascending order. `SortAndSearch()` should return the sorted list along with the number of comparisons required for sorting. What changes are required to arrange the elements of `Number_List` in descending order?

- Consider `Number_List = [2 15 5 4 12]` and `Num = 4`. Show the changes in `Number_List` after each iteration in `SortAndSearch()`.

Q6. Define a class `Product` that stores information about products manufactured by a company. The class should contain the following data members:

- `name`–Name of the `Product`
- `material`– Material with which `Product` is made of `'Metal'` or `'Plastic'`.
- `costPrice`– Cost of the `Product` in rupees. Include assert statement to ensure that the cost is between `25` and `250`.
- `discount`– Deduction in cost of the `Product`. It is equal to `20%` of `costPrice` if the material is `'Metal'` and `10%` for `'Plastic'`.
- `count` - number of objects created for `Product` class. Add appropriate statements in the code to increment the value of count by `1`, when an object is created. Use assert statement to check that count is always greater than equal to `0`.

The class should support the following methods:

- `__init__()` for initializing data members.
- `sellingPrice()` which computes and displays selling price where selling price is calculated as difference of `costPrice` and `discount`
- `display()` which displays the information about the product.

Also write Python statements for the following:

- Create a product `'Plate'` of `'Metal'` having `costPrice` as `170`. Display the value of `count`.
- Create a product `'Spoon'` of `'Plastic'` having `costPrice` as `26`. Display the value of `count`.
- Compute and display the selling price of `'Plate'` and `'Spoon'`. Display the value of `count`.