Unique Paper Code : 32345104

Name of the Paper : Programming using Python

Name of the Course : Computer Science: GE for Honours

Semester : I

Duration: 3 Hours Maximum Marks: 75

Instructions: Attempt any FOUR questions. All questions carry equal marks.

Q1

- Write an algorithm to find the least common multiple of two numbers.
- Which of the following are valid identifiers? If they are invalid, then mention the cause of violation.
    - `Nn3`
    - `pp_4`
    - `from`
    - `bv-1`
    - `highestOfHeight`
    - `2Good2BeTrue`
- Write a program to print the sum of the first `n` terms of the following series:

```
1/2 - 1/4 + 1/6 -1/8 + …
```

Q2

- Evaluate the following expressions and justify your answers:
    - `7 * 2 - 5 ** 2 // 4 + 8`
    - `10 < 5 or 7 < 12 and not 18 > 3`

- `float(6 - int(56.4) % 2)`
- `9 & -11`
- `'hello' < 'hi' or 'I am fine' > 'I am not fine'`

- Apply Selection Sort on the list `[95, 81, 14, 72, 68, 59, 42, 24]`, to sort it in the ascending order. Show the contents of the list after each iteration.

Q3 Write functions for the following:

- a function that accepts a number and a string as arguments. It should return a list having n elements such that the i^{th} element of the list has the string repeated i times. For example, if the arguments are `3` and `'Hello'` then the function should return `['Hello', 'HelloHello', 'HelloHelloHello']`. Use list comprehension to generate this list.
- a function that accepts two tuples as arguments and returns a third tuple which has the alternate elements from the two tuples passed to it as arguments. `IndexError` should be raised if the number of elements in the two tuples are not same. The function should have `try … except` statements to handle `IndexError`.
- a function that accepts a list and computes the frequency of each numeric element in the list. The non-numeric elements should be ignored. The function should return the answer as a dictionary. For example, if the list is `['Hello', 1.2, 5, [2, 7], 7, 5, 5, 1.2, 7.4]` then the output should be `{1.2: 2, 5: 3, 7: 1, 7.4: 1}`

Q4 What will be the output produced on execution of the following code segments? Justify your answers.

- ```
  s = 0
  for i in range(1, 10):
     if i % 2 == 0:
        continue
  ```
  ```

```
    else:
        s += i
print(s)
```

- ```
  color = set(['White', 'Green', 'Yellow', 'Blue'])
  primary = set(['Red', 'Green'])
  print('Green' in color)
  allcolor=color.union(primary)
  print(allcolor)
  print(color.intersection(primary))
  print(color.difference(primary))
  ```

- ```
  list1 = ['physics', 'chemistry', 1997, 2000]
  print(list1[1][-3:-9:-1])
  list1[2] = 2001
  del list1[2]
  print(list1[2])
  print(list1)
  ```

- ```
  txt = 'Hi there! Hello! Have a good day.'
  txt = txt.split('!')
  print(txt)
  print('!!!'.join(txt))
  ```

- ```
  def avg(marks):
      assert len(marks) != 0, "List is empty."
  ```

```
    return sum(marks)/len(marks)


try:
    mark1 = [55, 88, 78, 90, 79]
    print("Average of mark1:", avg(mark1))
    mark2 = []
    print("Average of mark2:", avg(mark2))
    print("Bye")
except AssertionError as err:
    print(err)
```

Q5

- A stack is initially empty. Show the contents of this stack on execution of each of the following operations: `push('Jan')`, `push('Feb')`, `pop()`, `push('Mar')`, `pop()`, `pop()`.

  What will happen if another `pop()` operation is invoked after the execution of the above sequence of operations?

- Write a function that accepts names of two files: source and target as arguments and copies alternate characters from the source file to the target file. The function should return -1 if the source file does not exist.

- Identify the errors in the following pieces of code and rewrite the code after removing them.

  ♦
```
DEF execmain():
   x = input("Enter a number:")
   if (abs(x) =  x):
      print("You entered a positive number")
   else:
```

```
x=*-1
    print "Number made positive:"x
```

♦
```
print('test' * 3.0)
print(3 + '5 '+ 7.0)
```

Q6 Define a class `Student` that keeps track of the academic record of students in a school. The class should contain the following instance data members:

`rollNo`: Roll number of the student

`name`: Student name

`classSection`: class and section of the student

`marksList`: List of marks in five subjects

The class should support following methods:

`__init__` for initializing the data members

`setMarks` to set marks for five subjects

`computeTotal` for returning the total of the marks

`__str__` that display information about an employee

Include assert statements in appropriate functions to ensure that the marks in the `marksList` are >= 0 and <= 100.

Also write statements for the following:

- create an object `S1` of this class for `Rajesh` of `IX C` having roll no. as 19 and marks as `92`, `96`, `83`, `97` and `91`.
- display the details of this student using `__str__` function