THE UNIVERSITY OF THE WEST INDIES

Department of Computing

COMP4620 - Programming Principles

Lab 1 - Python Basics

Question 1 (2)

Write a function **sum_z** that takes two parameters **num1** and **num2** and returns the sum of the two numbers.

```
>>> sum_z(5,6)
    11
```

Question 2 (2)

Write a function **check_num** that takes exactly one parameter **param**. The function should check if the **param** entered is an integer. If it is an integer the function should **return True else return False**.

Hint => You can use the function **isinstance** to check of the **param** is an integer.

```
>>> isinstance(3, int)
    True
>>> isinstance(3.1, int)
    False
```

```
>>> check_num(5.0)
    False
>>> check_num(5)
    True
```

Question 3 (3)

Write a function `sum_int_list` that takes a `list` as a parameter. The function should sum all the integers in the `list` and return the sum. This function must make use of the `check_num` function.

```
>>> lst = [3,4,1.4,6,'4','Word',7]
>>> sum_int_list(lst)
  20
```

Question 4 (3)

Write a function `future_date` that takes three parameters `year`, `month` and `day` (which represents a date) and returns `True` if the date is in future and `False` otherwise. These calculations are done by capturing the current date. The code given below captures the current date where the variables `current_year` contains the value for present year, `current_month` contains the value for the present month and `current_day` contains the value for the present day. Use these variables to check if the input date is in future.

```
import datetime
def future_date(yy,mm,dd):
     date = datetime.datetime.now()
     current_year = date.year
     current_month = date.month
     current_day = date.day
```

Question 5 (3)

Typically, programming languages have primitive `div` (i.e. /) and `mod` (i.e. %) operators. Assume that these operators do not exist in python, and you have to write your own functions. `div` takes two integers as input and keeps on subtracting the second from the first until the first number becomes less than the second number. The function keeps a track of how many times the second number is subtracted from the first and returns that number as the answer. `mod` also takes two integers as input and keeps on subtracting the second from the first until the first number becomes less than the second number. When the first number becomes less than the second, the value of the first number is the answer.

Write these two functions iteratively (using while loop only) in python:

i) **`div`** - division e.g. **`div`** `(11, 3) => 3`

ii) **`mod`** - remainder e.g. **`mod`** `(11, 3) => 2`

Question 6 (2)

Define a function **`isperfectSquare(n)`**, which returns **`True`** if n is a perfect square and **`False`** otherwise. Your function takes a positive integer as input and returns a Boolean Value. Use **`while`** loops in writing this function. A number n is a perfect square if there is a number from 1 to n-1 whose square is equal to n.

Examples of Perfect square:

```
9 = 3*3
16 = 4*4
```

```
>>> isperfectSquare(6)
    False
>>> isperfectSquare(9)
    True
```

Question 7 (5)

An integer greater than 1 is said to be prime if it divisible by only 1 and itself. For example 2,3,5,7 are prime numbers, but 4, 6, 8 and 9 are not. Write a function **`isPrime`** that determines whether a number is prime or not. The function takes a parameter n and checks if there exists a number from 2 to n-1 that n is divisible by **[Hint: you can use for loops].** If n is divisible by such a number then it is not a prime number and false must be returned. Return true if the number is a prime number. **Remember that 1 is not a prime number.**

```
isPrime(5) -> True
```

How to turn in this Lab

1. Save your python file.
2. Ensure that your file name follows this pattern:
   - Lab1_<*Your First Name*>_<*Your Last Name*>_<*Your ID Number*>.py
   - E.g. Lab1_Jane_Doe_642276142.py
3. Submit your single python (.py) file via the submission link on **OURVLE**.