Name : SK Shivaanee

Roll No : 2310257

Section : S02 **Assignment 6: RECURSIVE FUNCTIONS**

Aim: To learn python programming using recursive functions by forming expressions and statements involving reading and printing the data appropriately for the given specification.

Question: Write the Python code to solve the following problems using recursion. (CO3, K3) For each problem given,

- identify its input(s)
- identify the base case(s)
- state the recursive formula
- identify the return value of the recursive function
- write the pseudocode
- specify at least two test cases with sample input and output

Question 1: Finding the factorial of a number: Given a number n as input, find the value of n! . By definition, n !=n∗(n−1)∗(n−2)∗…∗1 In order to apply recursion, observe that the above equation is equivalent to n!=n∗(n−1)!

Input: The number whose factorial is to be found

Base Case:

```
if n==0 or n==1:
    return 1
```

Recursive formula : n*recursion(n-1)

Pseudocode:

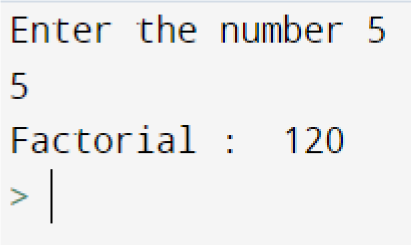
```
BEGIN
DECLARE FUNCTION recursion(n)
IF n==0 || n==1:
        RETURN 1
ELSE:
        RETURN n*recursion(n-1)
READ n
INVOKE r=recursion(n)
PRINT r
```

END

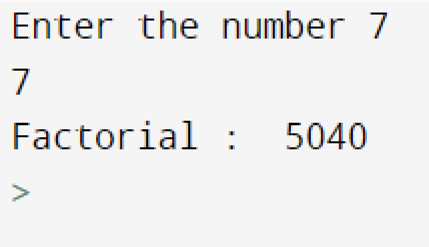
Source Code:

```
def recursion(n):
    if n==0 or n==1:
        return 1
    else:
        return n*recursion(n-1)
n=int(input("Enter the number"))
r=recursion(n)
print("Factorial : ",r)
```

OUTPUT:



```
Enter the number 5
5
Factorial :  120
> |
```



```
Enter the number 7
7
Factorial :  5040
>
```

Question 2: Finding the Greatest Common Divisor (GCD) of two numbers.

Input: Two numbers

Base Case:

```
if a == b:
    return a
```

Recursive Formula: gcd(b, a) and gcd(b, a - b)

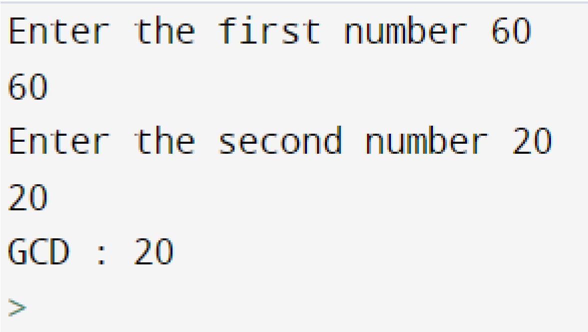
Pseudocode:

```
BEGIN
DECLARE FUNCTION gcd(a,b)
IF a==b:
RETURN a
ELIF a<b:
RETURN gcd(b,a)
ELSE:
RETURN gcd(b, a-b)
READ m and n
r=gcd(m,n)
PRINT r
```

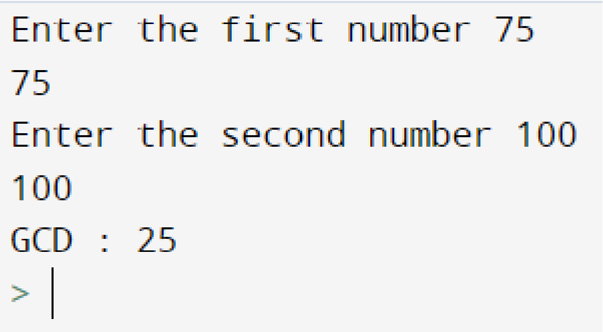
Source code:

```
def gcd(a, b):
  if a == b:
    return a
  elif a < b:
    return gcd(b, a)
  else:
    return gcd(b, a - b)
m=int(input("Enter the first number"))
n=int(input("Enter the second number"))
r=gcd(m,n)
print("GCD :",r)
```

OUTPUT:



```
Enter the first number 60
60
Enter the second number 20
20
GCD : 20
>
```



```
Enter the first number 75
75
Enter the second number 100
100
GCD : 25
>
```

Question 3: Finding the power of a number: Given a number n and an exponent p, find n p using the formula given below: power(n, p)=n∗power(n , p−1) You should use recursion, and NOT compute it directly using a for/while loop.

Input: The base and power values

Base case:

```
 if p==0:
      return 1
```

Recursive Formula: n*power(n,p-1)

Pseudocode:

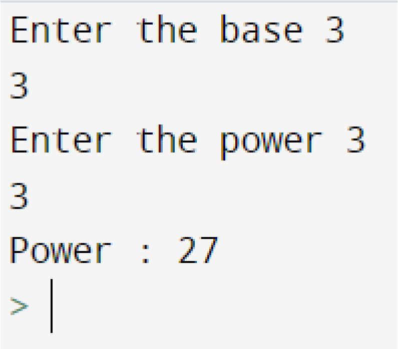
```
BEGIN
DECLARE FUNCTION power(n,p)
IF p==0:
        RETURN 1
```

```
ELSE:
    RETURN (n*power(n,p-1))
READ p and n
r=power(n,p)
PRINT r
```

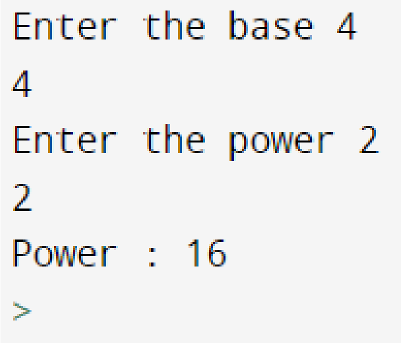
Source Code:

```
def power(n,p):
  if p==0:
    return 1
  else:
    return (n*power(n,p-1))
n=int(input("Enter the base"))
p=int(input("Enter the power"))
r=power(n,p)
print("Power :",r)
```

Output:



```
Enter the base 3
3
Enter the power 3
3
Power : 27
> |
```



```
Enter the base 4
4
Enter the power 2
2
Power : 16
>
```

Question 4: Find the sum of first n Fibonacci numbers: In mathematics, the Fibonacci sequence is a sequence in which each number is the sum of the two preceding ones. Numbers that are part of the Fibonacci sequence are known as Fibonacci numbers. (Source: Wikipedia) E.g. 0, 1, 1, 2, 3, 5, 8, 13, 21 are Fibonacci numbers.

Sample input: 5

Output: 7

Use the following recursive formulation to compute the n th Fibonacci number. fib(n)=fib(n−1)+fib(n−2)

Input: The number of terms

Base Case:

```
 if (n == 0):
     return 0
  if (n == 1):
     return 1
```

Recursive formula: fib_sum(n-1) + fib_sum(n-2) + 1

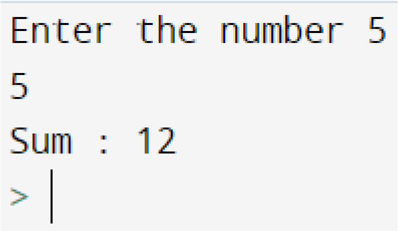
Pseudocode:

```
BEGIN

DECLARE FUNCTION fib_sum(n)

IF(n == 0):
    RETURN 0

ELIF(n == 1):
    RETURN 1
ELSE:
    RETURN fib_sum(n-1) + fib_sum(n-2) + 1

READ n

r=fib_sum(n)

PRINT r
```

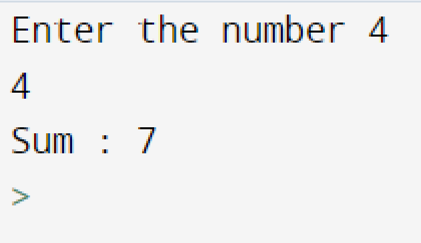
Source Code:

```
def fib_sum(n):
  if (n == 0):
    return 0
  if (n == 1):
    return 1
  else:
    return fib_sum(n-1) + fib_sum(n-2) + 1
n=int(input("Enter the number"))
r=fib_sum(n)
print("Sum :",r)
```

Output:



```
Enter the number 5
5
Sum : 12
> 
```



```
Enter the number 4
4
Sum : 7
>
```

Question 5: Find the sum of the digits of a number: Use a recursive formulation to find the sum of digits of a number. (Hint: Sum of digits of a number is given by the sum of its last digit plus the sum of digits of the number formed by all the preceding digits.)

Input: The number whose sum of digits is to be calculated

Base Case:

```
if l==1:
    return n
```

Recursive formula: n%10+sum_dig(n//10)

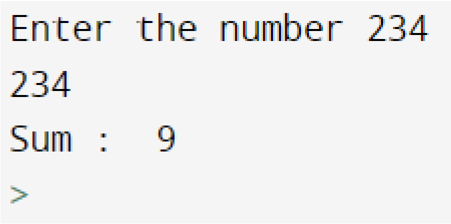
Pseudocode:

```
BEGIN
DECLARE FUNCTION sum_dig(n)
l=LEN(STR(n))
IF l==1:
        RETURN n
 ELSE:
      RETURN n%10+sum_dig(n//10)
READ n
r=sum_dig(n)
PRINT r
```

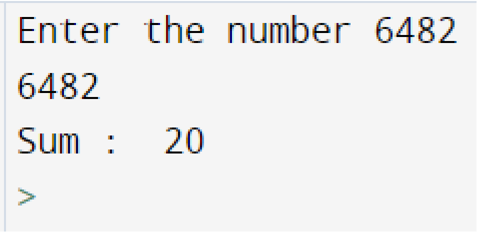
Source Code:

```
def sum_dig(n):
   l=len(str(n))
   if l==1:
      return n
   else:
      return n%10+sum_dig(n//10)
n=int(input("Enter the number"))
r=sum_dig(n)
print("Sum : ",r)
```

Output:



```
Enter the number 234
234
Sum :  9
>
```



```
Enter the number 6482
6482
Sum :  20
>
```

Question 6: Find the value of combinations: Given two numbers n and r, find n/ r using a recursive program. Recall that n /r = n!/(r!∗(n−r!))

Input: The values of n and r

Base Case:

```
if(n < r):
      return 0
   if(r == 0):
      return 1
   if(r == 1):
      return n
   if(n == 1):
      return 1
```

Recursive formula: comb(n - 1, r - 1) + comb(n - 1, r)

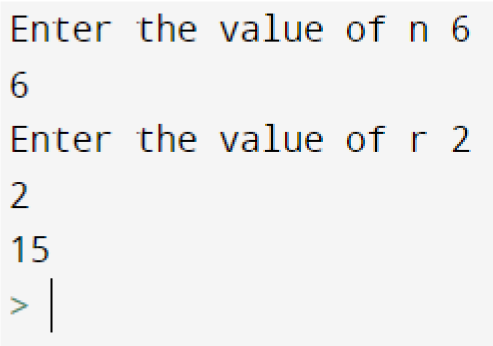
Pseudocode:

```
BEGIN

DECLARE FUNCTION comb(n,r)

 IF(n < r):
        RETURN 0
 IF(r == 0):
        RETURN 1
 IF(r == 1):
        RETURN n
 IF(n == 1):
        RETURN 1
    RETURN comb(n - 1, r - 1) + comb(n - 1, r)

READ n and r

PRINT comb(n,r)
```

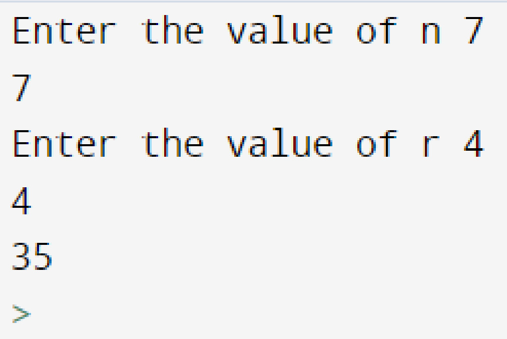
Source Code:

```
def comb(n, r):
    if(n < r):
        return 0
    if(r == 0):
        return 1
    if(r == 1):
        return n
    if(n == 1):
        return 1
    return comb(n - 1, r - 1) + comb(n - 1, r)
n=int(input("Enter the value of n"))
r=int(input("Enter the value of r"))
print(comb(n, r))
```

Output:



```
Enter the value of n 6
6
Enter the value of r 2
2
15
> 
```



```
Enter the value of n 7
7
Enter the value of r 4
4
35
>
```

Question 7: Checking for Armstrong number: A number is said to be an Armstrong number if the sum of its own digits raised to the power number of digits gives the number itself. Given a number n as input, print "Yes" if n is an Armstrong number, else print "No" if n is not an Armstrong number. (Hint: Use the same idea as the sum of the digits of a number in Qn 5)

Input: A number

Base Case:

```
 if num == 0:
      return num
```

Recursive formula:

pow((num%10),order) + check_armstrong(num//10)

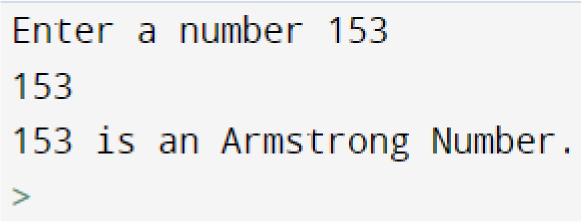
Pseudocode:

```
BEGIN

DECLARE FUNCTION check_armstrong(num):

 IF num == 0:
      RETURN num
 ELSE:
      RETURN pow((num%10),order) + check_armstrong(num//10)

READ num

order = len(str(num))
sum = check_armstrong(num)
IF sum == int(num):

         PRINT "is an Armstrong number"

ELSE:

         PRINT "is not an Armstrong number"
```

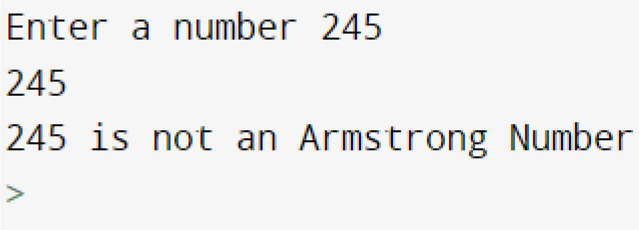
Source Code:

```
def check_armstrong(num):
   if num == 0:
      return num
   else:
      return pow((num%10),order) + check_armstrong(num//10)
num = int(input("Enter a number"))
order = len(str(num))
sum = check_armstrong(num)
if sum == int(num):
   print(num,"is an Armstrong Number.")
else:
   print(num,"is not an Armstrong Number")
```

Output:



```
Enter a number 153
153
153 is an Armstrong Number.
>
```



```
Enter a number 245
245
245 is not an Armstrong Number
>
```

Learning outcome:

1. Reading inputs / Printing the result

2. Using appropriate datatypes for the given input

3. Variable assignment

4. Converting the formula into python expressions

Result: Thus I learned to implement a simple problems in Python and solve the same using recursive functions.