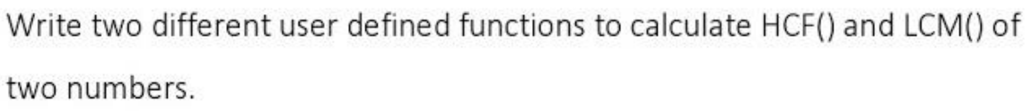
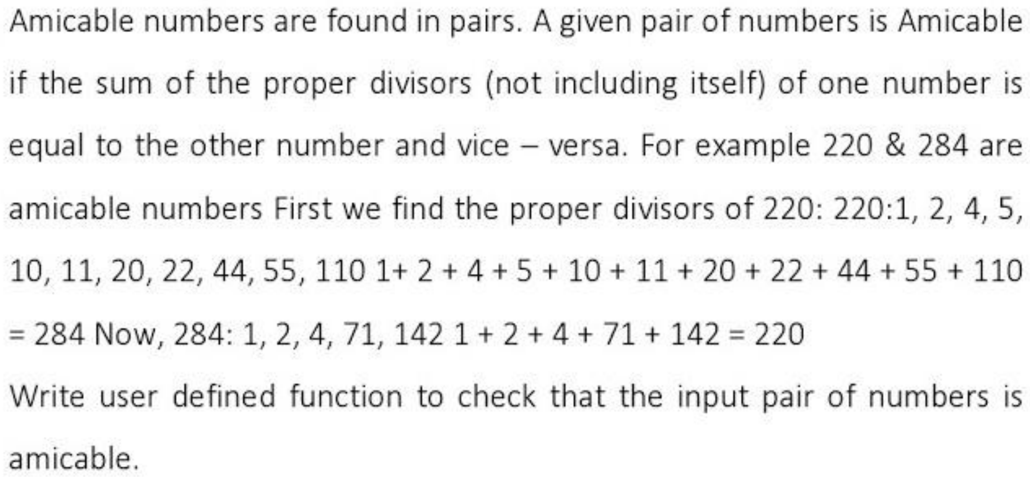
Exercise 1 – 

Write two different user defined functions to calculate HCF() and LCM() of two numbers.

Program –

```
def gcd(a,b):
    if(a>b):
        num=a
        deno=b
    else:
        num=b
        deno=a

    remainder = num%deno

    while(remainder!=0):
        num=deno
        deno=remainder
        remainder = num%deno

    gcd=deno
    return gcd

def lcm(a,b):
    lcm = a*b / gcd(a,b)
    print("GCD of {} & {} = {}".format(a,b,gcd(a,b)))
    print("LCM of {} & {} = {}".format(a,b,lcm))

flag=1

while(flag==1):
    a = int(input("Enter a number (a): "))
    b = int(input("Enter another number(b): "))
    lcm(a,b)

    intake=input("Do you wish to continue? (Press y to con
tinue or any key to exit.): ")

    if(intake=="Y" or intake=="y"):
        flag=1
    else:
        flag=0
        break
```

Exercise 2 –



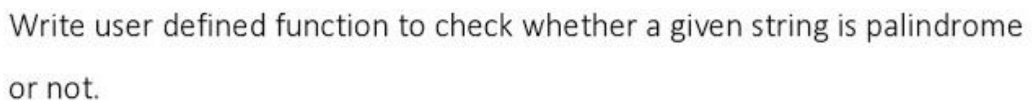
Amicable numbers are found in pairs. A given pair of numbers is Amicable if the sum of the proper divisors (not including itself) of one number is equal to the other number and vice – versa. For example 220 & 284 are amicable numbers First we find the proper divisors of 220: 220:1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 1+ 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284 Now, 284: 1, 2, 4, 71, 142 1 + 2 + 4 + 71 + 142 = 220

Write user defined function to check that the input pair of numbers is amicable.

Program –

```
def amicable(n1,n2):
    summ1=0
    summ2=0

    for i in range(1,(n1//2)+1):
        if(n1%i==0):
            summ1+=i

    for j in range(1,(n2//2)+1):
        if(n2%j==0):
            summ2+=j

    if((summ1==n2) and (summ2==n1)):
        print("Given numbers are Amicable")
    else:
        print("Given numbers are not Amicable")

flag=1

while(flag==1):
    n1=int(input("Enter 1st Number: "))
    n2=int(input("Enter 2nd Number: "))
    amicable(n1,n2)

    intake=input("Do you wish to continue? (Press y to continue or any key to exit.): ")

    if(intake=="Y" or intake=="y"):
        flag=1
    else:
        flag=0
        break
```

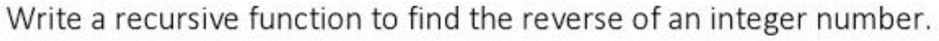
Exercise 3 –



Write user defined function to check whether a given string is palindrome or not.

Program –

```
def palindrome(word):
    p=0
    reverse = ""
    for i in word:
        reverse = i + reverse

    for i,j in zip(reverse,word):
        if(i==j):
            p=0
        elif(i!=j):
            p=1
            break

    if(p==0):
        print("The word is palindrome.")
    else:
        print("The word is not palindrome.")

flag=1

while(flag==1):
    word = input("Enter a word to check whether it is pali
ndrome or not: ")
    palindrome(word)

    intake=input("Do you wish to continue? (Press y to con
tinue or any key to exit.): ")

    if(intake=="Y" or intake=="y"):
        flag=1
    else:
        flag=0
        break
```

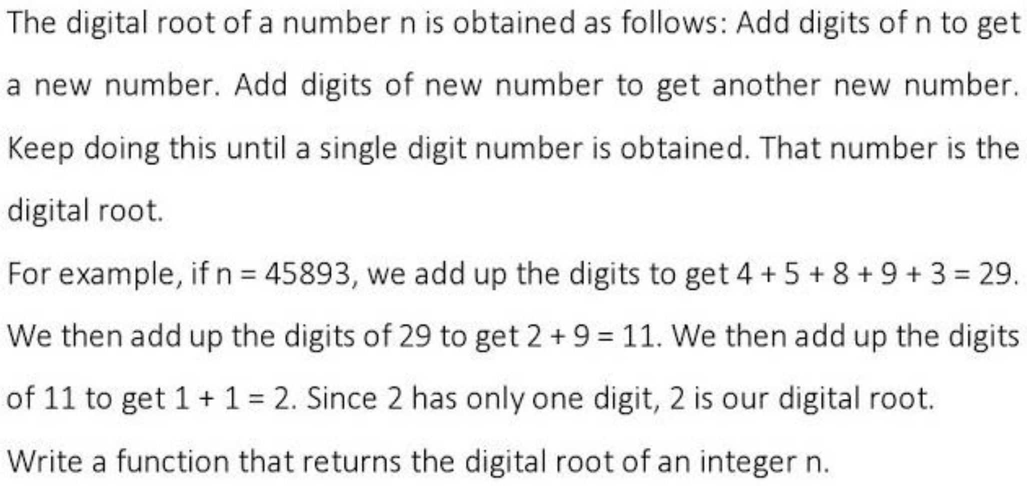
Exercise 4 – 

Write a recursive function to find the reverse of an integer number.

Program –

```
flag=1

while(flag==1):
    num=int(input("Enter number: "))
    summ=0
    def reverse(num):
        global summ
        if(num!=0):
            remainder=num%10
            summ=(summ*10)+remainder
            reverse(num//10)
        return summ

    print("Reverse is",reverse(num))

    intake=input("Do you wish to continue? (Press y to con
tinue or any key to exit.): ")

    if(intake=="Y" or intake=="y"):
        flag=1
    else:
        flag=0
        break
```

Exercise 5 –



The digital root of a number n is obtained as follows: Add digits of n to get a new number. Add digits of new number to get another new number. Keep doing this until a single digit number is obtained. That number is the digital root.

For example, if n = 45893, we add up the digits to get 4 + 5 + 8 + 9 + 3 = 29. We then add up the digits of 29 to get 2 + 9 = 11. We then add up the digits of 11 to get 1 + 1 = 2. Since 2 has only one digit, 2 is our digital root.

Write a function that returns the digital root of an integer n.

Program –

```
def digitalroot(n1,n2):
if(len(n2)==1):
     return n2
else:
     temp=n1
     summ=0
     while(temp!=0):
               remainder=temp%10
               summ+=remainder
               temp//=10
     n2=str(summ)
     return (digitalroot(summ,n2))

flag=1

while(flag==1):
     n1=int(input("Enter your Number: "))
     n2=str(n1)
     print("Digital root is",digitalroot(n1,n2))

     intake=input("Do you wish to continue? (Press y to co
     ntinue or any key to exit.): ")

     if(intake=="Y" or intake=="y"):
          flag=1
     else:
          flag=0
          break
```

Exercise 6 – Write a recursive function to check whether a given number is Armstrong or not.

Program –

```
flag=1

while(flag==1):
    num = int(input("Enter the number: "))
    size = len(str(num))
    sum = 0

    def isArmstrong(n):
        global sum
        global num
        global size
        if n < 10:
            sum += n ** size
            return num == sum
        else:
            sum += (n % 10) ** size
            return isArmstrong(n // 10)

    if isArmstrong(num):
        print(str(num) + " is an Armstrong Number.")
    else:
        print(str(num) + " is not an Armstrong Number.")

    intake=input("Do you wish to continue? (Press y to con
tinue or any key to exit.): ")

    if(intake=="Y" or intake=="y"):
        flag=1
    else:
        flag=0
        break
```