D D M M Y Y Y Y

EXERCISE-8

PROGRAM ON FUNCTIONS

Given two lists as arguments (marks and names), write a function to return a tuple containing the highest marks and the corresponding name.

```
def Highest(marks, name):
    if (len(marks) != len(name)):
        return "Number of records mismatch"
    max_marks = max(marks)
    index = []
    for i in range(len(marks)):
        if marks[i] == max_marks:
            index.append(i)
    res = []
    for i in index:
        res.append((marks[i], name[i]))
    return res

def enter(l, n, text = False):
    print("Enter the elements:")
    for i in range(n):
        if(text):
            l.append(input())
        else:
            l.append(int(input()))
```

Output:

```
Enter the number of marks : 3
Enter the elements
96
59
70
Enter the elements
Ajay
Shiv
Anvar
['96','Ajay']
```

```
marks, names = [], []
n1 = int(input("Enter the number of mark:"))
enter(marks, n1)
n2 = int(input("Enter the number of names:"))
enter(names, n2, True)
print(marks, names, sep="\n")
print(Highest(marks, names))
```

2. Given a dict where values are not unique, write a function to create a new dict where the key is the value and the value is concatenated keys of the original dict and return it.

```
def mod(d):
    res = {}
    for i in d:
        if d[i] not in res:
            res[d[i]] = []
            res[d[i]].append(i)
        else:
            res[d[i]].append(i)
    return res
x = {'apple': 'fruit', 'cat': 'mammal', 'beans': 'veg',
     'dog': 'mammal', 'mango': 'fruit', 'brinjal': 'veg',
     'potato': 'veg', 'horse': 'mammal'}
print(mod(x))
```

Output :

```
{'fruit' : ['mango', 'apple']
 'mammal' : ['cat', 'dog', 'horse']
 'veg' : ['beans', 'brinjal', 'potato'] }
```

D D M M Y Y Y Y

3. def is_square(x): pass
check whether a given number is a perfect square
def is_even(x): pass
check whether a given number is an even number
Find all the numbers between 1 and n which are both square and even.

```
def is_square(n):
    if int(n**0.5) == n**0.5:
        return True
    else:
        return False
def is_even(n):
    if n%2:
        return True
    else:
        return False
n = int(input("Enter upperlimit: "))
for i in range(2, n+1, 2):
    if is_even(i) and is_square(i):
        print(i, end = " ")
```

Output:

```
Enter upper limit : 36
 4
 16
 36
```

EXERCISE - 9

PROGRAM ON FILES AND MODULES

1. Given a file, create a new file with line numbers

```
f = open("Q1-Test.txt", 'r')
f1 = open("Q1-Output.txt", 'w')
for i,j in enumerate(f.readlines(), 1):
    f1.write(str(i) + " " + j)
f1.close()
f.close()
```

OUTPUT:

Q1_Test.txt:

Hello
Welcome to NMIT
Have a good day

Q1_Output.txt:

1. Hello
2 Welcome to NMIT
3 Have a good day

D D M M Y Y Y Y

2. Compare given two files
i) output union
ii) output intersection
iii) output those in file1 and not in file 2
iv) symmetric difference.

```
print("------ UNION ------")
with open('q2.txt') as f1, open('q1.txt') as f2:
    for i,j in zip(f1,f2):
        if set(i.split()) == set(j.split()):
            print(i.strip())
        else:
            print(i.strip(), j.strip())

print("---- INTERSECTION ----")
with open('q2.txt') as f1, open('q1.txt') as f2:
    for i,j in zip(f1,f2):
        if set(i.split()) == set(j.split()):
            print(i.strip())

print("------ set1 - set2 ------")
with open('q2.txt') as f1, open('q1.txt') as f2:
    for i,j in zip(f1,f2):
        if set(i.split()) == set(j.split()):
            pass
        else:
            print(i.strip())
```

OUTPUT:

q1.txt

Hello
Welcome to NMIT
Please collect your ID card in the counter

q2.txt

Hello
Have a good day

// Output:

---- UNION ----

Hello
Have a good day welcome to NMIT

----- INTERSECTION -----

Hello

----- set1 - set2 -----

Have a good day

D D M M Y Y Y Y

```
print("------SYMMETRIC DIFFERENCE------")
with open('q2.txt') as f1, open('q1.txt') as f2:
    for i,j in zip(f1,f2):
        if set(i.split())==set(j.split()):
            pass
        else:
            print(i.strip(),j.strip())
```

- - - - SYMMETRIC DIFFERENCE - - - - -

Have a good day welcome to NMIT

D D M M Y Y Y Y

3. Input file is a python program with function definitions.
i) Identify the leaders and write them to another file
ii) Identify the functions and print them.

```
out = open('leaders.txt', 'w')
out.writelines("Leaders are \n Line No \t
                    Code Snippet \n")
with open("9b.py", 'r') as IP:
    for j, i in enumerate(IP, 1):
        if i.strip().endswith(":"):
            out.writelines(str(j) + '\t' + i.strip()
                           [:-1] + '\n')
out.close()

out = open("Functions.txt", 'w')
out.writelines("Functions are :\n Line no \t
                    Line of code\n")
with open("9b.py", 'r') as IP:
    for i, j in enumerate(IP, 1):
        if j.strip().endswith("def"):
            out.writelines(str(i) + '\t' + j.strip()
                           [:-1] + '\n')
out.close()
```

D D M M Y Y Y Y

4. Create a module called util.py. Add functions for the following into this file.

a) Convert temperature in Centigrade to Fahrenheit.

b) Convert temperature in Fahrenheit to Centigrade.

```
s = """
def c_to_f(Celsius):
    Farenheit = ((9/5) * Celsius) + 32
    return Fareheit
def f_to_c(Farenheit):
    Celsius = ((5/9) * Farenheit - 32))
    return Celsius
"""
with open("util.py", 'x+') as Module:
    Module.writelines(s)
```

Output:

util.py:

```
def c_to_f(Celsius):
    Farenheit = ((9/5) * Celsius) + 32
    return Farenheit
def f_to_c(Farenheit):
    Celsius = ((5/9) * (Farenheit - 32))
    return Celsius
```

5. Create a module called MyStat.py. Support functions:
- sum - average - standard deviation

```
import util as U
import MyStat as MS
c = int(input("Enter temp in celsius:"))
print(U.c_to_f(c))
f = int(input("Enter temp in farenheit:"))
print(U.f_to_c(f))

l = []
n = int(input("Enter size of list:"))
print("Enter elements:")
for i in range(n):
    l.append(int(input()))

print(l)
print("Sum of all elements:", MS.Sum(l))
print("Average/mean:", MS.Average(l))
print("Standard deviation:", MS.Standard_
                              Deviation(l))
```