Lab 4 Python Basics

Name: Ketan Goud
Reg No: 220905260
Roll No: 39
Section: CSE D D2

1. Write a python program to reverse a content a file and store it in another file.

```
f1=open("file1.txt","rt")
f2=open("file2.txt","wt")
print(f1.readline())
with open("file1.txt","rt") as my_file:
    data=my_file.read()
rev_data=data[::-1]
f2.write(rev_data)
f2.close()
f2=open("file2.txt","rt")
print(f2.readline())
```

[15] ✓ 0.0s

```
Hello World!
!dlroW olleH
```

2. Write a python program to implement binary search with recursion.

```
def binary_search(arr,low,high,x):
    if(high>=low):
        mid=(low+high)//2
        if(arr[mid]==x):
            return mid
        elif(arr[mid]>x):
            return binary_search(arr,low,mid-1,x)
        else:
            return binary_search(arr,mid+1,high,x)
    else:
        return -1

arr=[10,20,30,40,50,60]
x=40
res=binary_search(arr,0,len(arr)-1,x)
if(res!=-1):
    print(f"The number {x} is at the index {res}")
else:
    print(f"The number {x} is not there in the array")
```

[19] ✓ 0.0s

```
The number 40 is at the index 3
```

3. Write a python program to sort words in alphabetical order.

```
my_str=str(input("Enter your string:"))
words=[word.lower() for word in my_str.split()]
words.sort()
for word in words:
    print(word)
```

22] ✓ 7.3s

```
am
goud
hello
i
ketan
```

4. Write a Python class to get all possible unique subsets from a set of distinct integers Input:[4,5,6]
Output : [[], [6], [5], [5, 6], [4], [4, 6], [4, 5], [4, 5, 6]]

```
class SubsetGenerator:
    def __init__(self, nums):
        self.nums = nums
        self.result = []

    def generate_subsets(self):
        self._backtrack(0, [])
        return self.result

    def _backtrack(self, start, current):
        self.result.append(current[:])
        for i in range(start, len(self.nums)):
            current.append(self.nums[i])
            self._backtrack(i + 1, current)
            current.pop()

nums = [4, 5, 6]
subset_generator = SubsetGenerator(nums)
result = subset_generator.generate_subsets()
print(result)
```

[32] ✓ 0.0s

```
[[], [4], [4, 5], [4, 5, 6], [4, 6], [5], [5, 6], [6]]
```

5. Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.
Input: numbers= [10,20,10,40,50,60,70], target=50
Output: 3, 4.

```
class pair_sum:
    def __init__(self, numbers, target):
        self.numbers=numbers
        self.target=target
    def find_pairs(self):
        visited={}
        pair_indices={}
        for i,num in enumerate(self.numbers):
            complement=self.target-num
            if(complement in visited):
                return (visited[complement],i)
            visited[num]=i
        return None

numbers = [10,20,30,40,50,60,70,80]
target=50
p=pair_sum(numbers,target)
res=p.find_pairs()
if(res):
    print(f"Pair found at {res}")
else:
    print("No pair found")
```

31] ✓ 0.0s

```
Pair found at (1, 2)
```

6. Write a Python class to implement pow(x, n).

```
class power_func:
    def __init__(self,x,n):
        self.x=x
        self.n=n
    def pow(self):
        return self.x**self.n

p=power_func(5,2)
print(p.pow())
p=power_func(10,4)
print(p.pow())
```

[29] ✓ 0.0s

```
25
10000
```

7. Write a Python class which has two methods get_String and print_String. The get_String accept a string from the user and print_String print the string in upper case.

```
class strings:
    def get_string(self):
        self.my_str=str(input("Enter your string:"))
    def print_string(self):
        print(self.my_str)

st=strings()
st.get_string()
st.print_string()
```

[25] ✓ 3.6s

```
Hello
```