Practical 1

AIM :- To implement basic operations on a one-dimensional array such as traversal, insertion, deletion and searching.

1-D Array Operation

Code :-

```
arr = [ ]
n = int (input ("Enter number of elements: "))
for i in range (n):
    element = int (input ("Enter Element: "))
    arr. append (element)

print ("Array Element are : ")
for i in arr:
    print (i)

arr. apend (100)
print. remove
print (" array insertion : ", arr)

arr. remove (arr [0])
print (" After deletion : ", arr)
```

P-1.1-D Array OUTPUT

```
Enter number of elements: 5
Enter Element: 7
Enter Element: 2
Enter Element: 8
Enter Element: 3
Enter Element: 9
Array Element are:
7
2
8
3
9
Array insertion: [7, 2, 8, 3, 9, 100]
After deletion: [2, 8, 3, 9, 100]
Enter element to search: 7
Element not found
```

```
search = int(input("Enter element to search: "))
if search in arr:
    print("Element found")
else:
    print("Element not found")
```

P-1. 2-D Array OUTPUT

```
Enter number of rows : 3
Enter number of Columns : 3
Enter element [0][0] : 5
Enter element [0][1] : 6
Enter element [0][2] : 4
Enter element [1][0] : 7
Enter element [1][1] : 8
Enter element [1][2] : 2
Enter element [2][0] : 9
Enter element [2][1] : 1
Enter element [2][2] : 4

2-D Array elements are :
5  6  4
7  8  2
9  1  4
```

P-1.2-D array

AIM:- To implement a two-diamensional array and display its elements using nested loops:

Code :-

```
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))
matrix = []
for i in range(rows):
    row = []
    for j in range(cols):
        element = int(input(f"Enter element [{i}][{j}]: "))
        row.append(element)
    matrix.append(row)

print("2-D Array elements are :")
for i in range(rows):
    for j in range(cols):
        print(matrix[i][j], end=" ")
    print()
```

Practical 2

AIM:- To create a stack using Python list and perform stack operations such as push, pop, peek and display.

CODE:-

```
stack = [ ]
while True:
    pyth print ("\n-- Stack Menu --")
    print ("1- Push")
    print (" 2. Pop")
    print (" 3. Peek")
    print (" 4. Display")
    print ("5. Exit")

    choice = int (input ("Enter Your choice : "))

    if choice == 1:
        item = int (input ("Enter element to push : "))
        stack.append (item)
        print ("Element Pushed")

    elif choice == 2:
        if len (stack) == 0:
            print ("stack is empty")
        else:
            removed = stack.pop ()
            print ("Popped element :", removed)
```

Practical - 2 OUTPUT

```
-- Stack Menu --
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your Choice: 1
Enter element to push: 5
Element Pushed

-- Stack Menu --
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your Choice: 4
Stack element: [5]
```

```
elif Choice == 3:
    if len(stack) == 0:
        print("stack is empty")
    else:
        print("Top element: ", stack[-1])

elif choice == 4:
    if len(stack) == 0:
        print("stack is empty")
    else:
        print("Stack element: ", stack)

elif choice == 5:
    print("Exiting Program")
    break
else:
    print("Invalid choice")
```

Practical 3

AIM :- To implement different sorting algorithm such as Bubble Sort, Selection Sort, and Insertion Sort using Python.

P-3.1 Bubble ~~Selection~~ Sort

CODE :-

```
arr = [5, 1, 4, 2, 8]
n = len(arr)
for i in range(n):
    for j in range(0, n-i-1):
        if arr[j] > arr[j+1]:
            arr[j], arr[j+1] = arr[j+1, arr[j]
```

```
print("Sorted array using Bubble Sort:", arr)
```

P-3.2 Selection Sort

CODE :-

```
arr = [64, 25, 12, 22, 11]
n = Len(arr)
for in in range(n):
    min_index = i
    for j in range(i+1, n):
        if arr[j] < arr[min_index]:
            min_index = j
    arr[i], arr[min_index] = arr[min_index], arr[i]
```

Practical - 3 OUTPUT

P-3.1 Bubble Sort

Sorted array using Bubble Sort: [1, 2, 4, 5, 8]

P-3.2 Selection Sort

Sorted array using Selection Sort: [11, 12, 22, 25, 64]

P-3.3 Insertion Sort

Sorted array using Insertion Sort: [5, 6, 11, 12, 13]

```
print("Sorted array using selection Sort:", arr)
```

P-3.3 Insertion Sort

CODE :-

```
arr = [12, 11, 13, 5, 6]
for i in range (1, len(arr)):
    key = arr[i]
    j = i - 1
    while j >= 0 and key < arr[j]:
        arr[j+1] = arr[j]
        j -= 1
        arr[j+1] = key

print("Sorted array using Insertion sort:", arr)
```

Practical 4

AIM :- To implement recursive algorithms using Python such as Factorial, Fibonacci series, and Tower of Hanoi.

P-4.1 Factorial using Recursion

CODE :-

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n-1)

num = int(input("Enter a number: "))
print("Factorial is: ", factorial(num))
```

Practical - 4 OUTPUT

P-4.1 Factorial using Recursion

```
Enter a number : 5
Factorial is : 120
```

P-4.2 Fibonacci using Recursion

```
Enter number of term : 3
fibonacci series: 011
```

P- 4.3 Tower of Hanoi using Recursion

```
Enter number of disk : 3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

4.2 Fibonacci using Recursion

CODE:-

```
def fibonacci(n):
    if n<=1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)
term = int(input("Enter number of term: "))
print("Fibonacci Series: ")
for i in range(term):
    print(fibonacci(i), end = " ")
```

4.3 Tower of Hanoi using Recursion

CODE:-

```
def toh(n, source, auxiliary, destination):
    if n==1:
        print("Move disk 1 from", source, "to", destination)
        return
    toh(n-1, source, destination, auxiliary)
    print("Move disk", n, "from", source, "to", destination)
    toh(n-1, auxiliary, source, destination)

disks = int(input("Enter number of disk:"))
toh(disks, 'A', 'B', 'C')
```

Practical 5 OUTPUT

Linear Search Result: 2

Practical 5

AIM:- To write a Python program that perform linear search to find a given element in a list and returns its positions

CODE:-

```
def linear_search(data, target):
    for i in range(len(data)):
        if data[i] == target:
            return i
    return "NOT FOUND"
my_list = [5, 2, 9, 1, 7]
print(f"Linear Search Result: {linear_search(my_list, 9)}")
```

Practical - 6 OUTPUT

Fibonacci Sequence:

0
1
1
2
3
5
8
13
21
34

Practical 6

AIM:- To write a program fibonacci series using dynamic programming.

CODE:-

```
def recur_fibo(n):
    if n<=1:
        return n
    else:
        return (recur_fibo(n-1)+recur_fibo(n-2))
nterms = 10
if nterms<=0:
    print("Please enter a position integer")
else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print(recur_fibo(i))
```

Practical - 7 OUTPUT

List: [10, 4, 7, 2, 9, 15, 6]

2th Minimum Element: 4

2th Maximum Element: 10

Practical 7

AIM :- Write a python program to find the nth maximum and nth minimum elements.

CODE :-

```
numbers = [10, 4, 7, 2, 9, 15, 6]
n = 2
unique_numbers = list(set(numbers))
unique_numbers.sort()
nth_min = unique_numbers[n-1]
nth_max = unique_numbers[-2]
print("List:", numbers)
print(f"{n}th Minimum Element:", nth_min)
print(f"{n}th Maximum Element:", nth_max)
```

Practical - 8 OUTPUT

Pattern found in the String

Practical - 8

AIM :- To write a python program to find a pattern in given string.

CODE :-

```
String = "Python Programming in easy"
Pattern = "Programming"
if Pattern in string:
    print("Pattern found in the String.")
else:
    print("Pattern not found in the String.")
```