**PRACTICAL NO.01**

**A)To Implement 1-D Array.**

**Method-1 : Direct Method**

**Code:-**

import numpy as np

array\_1d = np.array([10, 20, 30, 40, 50])

print("1-D Array:")

print(array\_1d)

print("\nFirst element:", array\_1d[0])

print("Last element:", array\_1d[-1])

array\_1d[2] = 100

print("\nModified Array:")

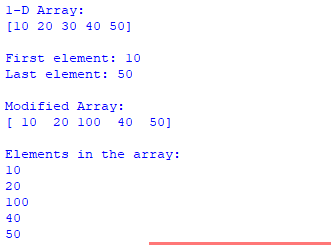
print(array\_1d)

print("\nElements in the array:")

for element in array\_1d:

print(element)

**Output:-**



**B)To Implement 2-D Array.**

**Code:-**

import numpy as np

array\_2d=np.array([[1,2,3],

[4,5,6],

[7,8,9]])

print("2-D Array:")

print(array\_2d)

print("\nElement at (1,2):", array\_2d[1,2])

print("First row:",array\_2d[0])

array\_2d[0,1]=99

print("\nModified Array:")

print(array\_2d)

print("\nElements in the 2-D array:")

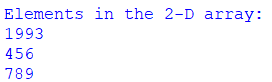
for row in array\_2d:

for elements in row:

print(elements,end='')

print()

**Output:-**

****

**PRACTICAL NO.02**

**Aim:-Create a list-based stacks and performs stack operations.**

**Code:-**

class Stack:

def \_\_init\_\_(self):

self.items = []

def is\_empty(self):

return len(self.items) == 0

def push(self, item):

self.items.append(item)

def pop(self):

if self.is\_empty():

raise IndexError("pop from empty stack")

return self.items.pop()

def peek(self):

if self.is\_empty():

raise IndexError("peek from empty stack")

return self.items[-1]

def size(self):

return len(self.items)

def display(self):

print("Stack:", self.items)

if \_\_name\_\_ == "\_\_main\_\_":

stack = Stack()

stack.push(10)

stack.push(20)

stack.push(30)

stack.display()

print("Top item:", stack.peek())

print("Popped item:", stack.pop())

stack.display()

print("Is stack empty?", stack.is\_empty())

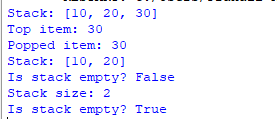
print("Stack size:", stack.size())

stack.pop()

stack.pop()

print("Is stack empty?", stack.is\_empty())

**Output:-**



**PRACTICAL NO.03**

**Aim: Implement linear and binary search algorithms on a list.**

**Code:-**

def linear\_search(arr, target):

for index, element in enumerate(arr):

if element == target:

return index

return -1

def binary\_search(arr, target):

beg = 0

end = len(arr) - 1

while beg <= end:

mid = (beg + end) //2

mid\_val = arr[mid]

if mid\_val == target:

return mid

elif target < mid\_val:

end = mid-1

else:

beg = mid+1

return -1

if \_\_name\_\_ == "\_\_main\_\_":

nums = list(map(int, input("Enter numbers separated by spaces: ").split()))

target1 = int(input("Enter the number to search for: "))

linear\_result = linear\_search(nums, target1)

if linear\_result != -1:

print(f"Linear Search: Found {target1} at index {linear\_result} in the original list.")

else:

print(f"Linear Search:{target1} not found.")

sorted\_nums = sorted(nums)

binary\_result = binary\_search(sorted\_nums, target1)

if binary\_result != -1:

print(f"Binary Search: Found {target1} at index {binary\_result} in sorted list {sorted\_nums}.")

else:

print(f"Binary Search: {target1} not found in sorted list {sorted\_nums}.")

**Output:-**



**PRACTICAL NO.04**

**Aim:-**

**A)To Implement Bubble Sort array**

**Code:-**

def bubble\_sort(arr):

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]

return arr

if \_\_name\_\_ == "\_\_main\_\_":

user\_input = input("Enter a list of numbers {separated by spaces}: ")

arr= list(map(int, user\_input.split()))

print("Before sorting: ", arr)

arr = bubble\_sort(arr)

print("After sorting: ", arr)

Output:-



**B)To Implement Selection sort array**

**Code:-**

def selection\_sort(arr):

n=len(arr)

for i in range(n-1):

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]

return arr

if \_\_name\_\_=="\_\_main\_\_":

user\_input=input("Enter a list of numbers (separated by spaces):")

arr=list(map(int, user\_input.split()))

print("Before sorting:",arr)

arr=selection\_sort(arr)

print("After sorting:",arr)

**Output:-**



**C)To Implement insertion sort array**

**Code:-**

def insertion\_sort(arr):

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

return arr

if \_\_name\_\_=="\_\_main\_\_":

user\_input=input("Enter a list of numbers(separated by spaces):")

arr=list(map(int,user\_input.split()))

print("Before sorting:",arr)

arr=insertion\_sort(arr)

print("After sorting:",arr)

**Output:-**



**PRACTICAL NO.05**

**Aim:-Implement algorithm to find Nth Max/Min element in a list**

**Code:-**

import random

def partition\_min(nums, left, right, pivot\_index):

pivot\_value=nums[pivot\_index]

nums[pivot\_index], nums[right] = nums[right], nums[pivot\_index]

store\_index = left

for i in range(left, right):

if nums[i] < pivot\_value:

nums[store\_index], nums[i]=nums[i], nums[store\_index]

store\_index+=1

nums[right], nums[store\_index] = nums[store\_index], nums[right]

return store\_index

def partition\_max(nums, left, right, pivot\_index):

pivot\_value=nums[pivot\_index]

nums[pivot\_index], nums[right]=nums[right], nums[pivot\_index]

store\_index=left

for i in range(left, right):

if nums[i] > pivot\_value:

nums[store\_index], nums[i]=nums[i], nums[store\_index]

store\_index+=1

nums[right], nums[store\_index]= nums[store\_index], nums[right]

return store\_index

def quickselect(nums, left, right, n, order):

if left==right:

return nums[left]

pivot\_index=random.randint(left, right)

if order=='min':

pivot\_index=partition\_min(nums, left, right, pivot\_index)

else:

pivot\_index=partition\_max(nums, left, right, pivot\_index)

if n==pivot\_index:

return nums[n]

elif n <pivot\_index:

return quickselect(nums, left, pivot\_index-1, n, order)

else:

return quickselect(nums, pivot\_index + 1, right, n, order)

def nth\_element(nums, n, order='min'):

if n>len(nums):

return None

order\_type='min' if order == 'min' else 'max'

return quickselect(nums, 0, len(nums) - 1, n-1, order\_type)

if \_\_name\_\_=="\_\_main\_\_":

user\_input = input("Enter a list of numbers (separated by spaces): ")

nums=list(map(int, user\_input.split()))

n=int(input("Enter the value of N (for Nth min/max):"))

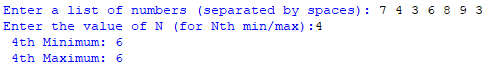
nth\_min = nth\_element(nums, n, order='min')

print(f" {n}th Minimum: {nth\_min}")

nth\_max = nth\_element(nums, n, order='max')

print(f" {n}th Maximum: {nth\_max}")

**Output:-**



**PRACTICAL NO.06**

**Aim:-**

**A) Implement Naïve String algorithm/ Brute Force algorithm to find a pattern in a given string.**

**Code:-**

def naive\_string\_match(text, pattern):

text\_len=len(text)

pattern\_len=len(pattern)

for i in range(text\_len-pattern\_len+1):

if text[i:i+pattern\_len]==pattern:

print(f"pattern found at index {i}")

if \_\_name\_\_=="\_\_main\_\_":

text="ababcabcabababd"

pattern="abab"

naive\_string\_match(text,pattern)

**Output:-**



**B) Implement KMP algorithm to find a pattern in a given string.**

**Code:-**

def compute\_lps(pattern):

lps=[0]\*len(pattern)

length=0

i=1

while i<len(pattern):

if pattern[i]==pattern[length]:

length+=1

lps[i]=length

i+=1

else:

if length !=0:

length=lps[length-1]

else:

lps[i]=0

i+=1

return lps

def kmp\_string\_match(text, pattern):

lps=compute\_lps(pattern)

i=j=0

while i<len(text):

if pattern[j]==text[i]:

i+=1

j+=1

if j==len(pattern):

print(f"pattern found at index {i-j}")

j=lps[j-1]

elif i<len(text) and pattern[j] !=text[i]:

if j !=0:

j=lps[j-1]

else:

i+=1

if \_\_name\_\_=="\_\_main\_\_":

print("Author: Rajesh Yadav")

text="ababcabcabababd"

pattern="abab"

kmp\_string\_match(text,pattern)

**Output:-**



**PRACTICAL NO.07**

**Aim:-**

**A)Implement factorial using recursion**

**Code:-**

def factorial(n):

if n==0 or n==1:

return 1

return n\*factorial(n-1)

if \_\_name\_\_=="\_\_main\_\_":

print("Author: Rajesh Yadav")

n=int(input("Enter the value of n to find the factorial of number:"))

print(f"The factorial of {n} is: {factorial(n)}")

**Output:-**



**B) Implement fibonicci series using recursion**

**Code:-**

def fibonacci(n):

if n==0:

return 0

elif n==1:

return 1

return fibonacci(n-1)+fibonacci(n-2)

if \_\_name\_\_=="\_\_main\_\_":

print("Author: Rajesh Yadav")

n=int(input("Enter the value of n to find the fibonacci of number:"))

print(f"The {n}th Fibonacci number is: {fibonacci(n)}")

**Output:-**



**PRACTICAL NO.08**

**Aim:- File merging using the Greedy Algorithm.**

**Code:-**

def merge\_files(file\_sizes):

total\_cost=0

while len(file\_sizes)>1:

file\_sizes.sort()

first\_smallest=file\_sizes.pop(0)

second\_smallest=file\_sizes.pop(0)

merged\_size=first\_smallest+second\_smallest

total\_cost+=merged\_size

file\_sizes.append(merged\_size)

return total\_cost

if \_\_name\_\_=="\_\_main\_\_":

print("Enter the sizes of the files separated by spaces:")

sizes\_input=input()

file\_sizes=list(map(int, sizes\_input.split()))

total\_merge\_cost = merge\_files(file\_sizes)

print(f"The total cost of merging the files is: {total\_merge\_cost}")

**Output:-**



**B) Coin change using the Greedy Algorithm.**

**Code:-**

def greedy\_coin\_change(amount, denominations):

denominations.sort(reverse=True)

coins\_used={}

for coin in denominations:

count=amount//coin

if count>0:

coins\_used[coin]=count

amount-=count\*coin

return coins\_used

if \_\_name\_\_=="\_\_main\_\_":

print("Enter the coin denominations separated by spaces:")

coins\_input=input()

coins=list(map(int, coins\_input.split()))

print("Enter the amount to change:")

amount=int(input())

coins\_used=greedy\_coin\_change(amount, coins)

total\_coins=sum(coins\_used.values())

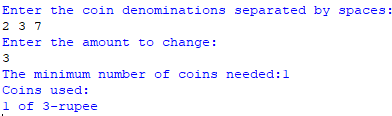
print(f"The minimum number of coins needed:{total\_coins}")

print("Coins used:")

for coin, count in coins\_used.items():

print(f"{count} of {coin}-rupee")

**Output:-**



**PRACTICAL NO.09**

**Aim:- Implement algorithms like merge sort using divide conquer.**

**Code:-**

def merge(left, right):

merged = []

i = j = 0

while i < len(left) and j < len(right):

if left[i] < right[j]:

merged.append(left[i])

i += 1

else:

merged.append(right[j])

j += 1

merged.extend(left[i:])

merged.extend(right[j:])

return merged

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left\_half = merge\_sort(arr[:mid])

right\_half = merge\_sort(arr[mid:])

return merge(left\_half, right\_half)

if \_\_name\_\_ == "\_\_main\_\_":

user\_input = input("Enter numbers separated by commas: ")

array = list(map(int, user\_input.split(',')))

sorted\_array = merge\_sort(array)

print("Sorted array:", sorted\_array)

**Output:-**



**B) Implement Strassen’s Matrix Multiplication Algorithm using divide and conquer.**

**Code-**

import numpy as np

def strassen(A, B):

if A.shape[0] == 1:

return A \* B

n = A.shape[0]

mid = n // 2

A11=A[:mid, :mid]

A12=A[:mid, mid:]

A21=A[mid:, :mid]

A22=A[mid:, mid:]

B11=B[:mid, :mid]

B12=B[:mid, mid:]

B21=B[mid:, :mid]

B22=B[mid:, mid:]

P1 = strassen(A11, B12 - B22)

P2 = strassen(A11 + A12, B22)

P3 = strassen(A21 + A22, B11)

P4 = strassen(A22, B21 - B11)

P5 = strassen(A11 + A22, B11 + B22)

P6 = strassen(A12 - A22, B21 + B22)

P7 = strassen(A11 - A21, B11 + B12)

C11 = P5 + P4 - P2 + P6

C12 = P1 + P2

C21 = P3 + P4

C22 = P5 + P1 - P3 - P7

C = np.zeros((n, n), dtype=A.dtype)

C[:mid, :mid] = C11

C[:mid, mid:] = C12

C[mid:, :mid] = C21

C[mid:, mid:] = C22

return C

if \_\_name\_\_ == "\_\_main\_\_":

print("Author: Rajesh Yadav")

print("Enter matrix A (rows separated by semicolons, values separated by commas):")

A\_input = input()

A = np.array([list(map(int, row.split(','))) for row in A\_input.split(';')])

print("Enter matrix B (rows separated by semicolons, values separated by commas):")

B\_input = input()

B = np.array([list(map(int, row.split(','))) for row in B\_input.split(';')])

print("Original Matrix A:")

print(A)

print("Original Matrix B:")

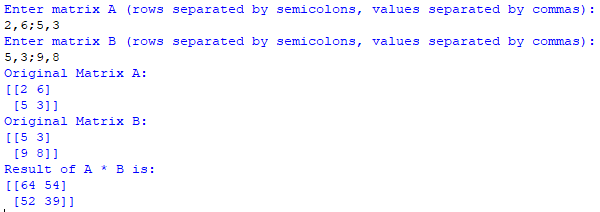
print(B)

C = strassen(A, B)

print("Result of A \* B is:")

print(C)

**Output:-**



**PRACTICAL NO.10**

**Aim:- Fibonacci Series using dynamic programming.**

**Code:-**

def fibonacci(n):

if n <= 0:

return 0

elif n == 1:

return 1

fib = [0] \* (n + 1)

fib[0], fib[1] = 0, 1

for i in range(2, n + 1):

fib[i] = fib[i - 1] + fib[i - 2]

return fib[n]

if \_\_name\_\_ == "\_\_main\_\_":

print("Author: Rajesh Yadav")

n = int(input("Enter the value of n to find the nth Fibonacci number: "))

print(f"The {n}th Fibonacci number is: {fibonacci(n)}")

**Output:-**

