**Q. 1) Write a Python Program to Calculate the Average of Numbers in a Given List.**

num=[10,20,30,40]

avg=sum(num)/len(num)

print("avg of list element:",avg)

**Q.2) Implement searching algorithms to search an element using: Linear Search**

def linear\_search(arr, target):

for i in range(len(arr)):

if arr[i] == target:

return f"Element {target} found at index {i}."

return f"Element {target} not found in the list."

my\_list = [10, 23, 45, 7, 15, 28, 32]

element\_to\_find = 15

result = linear\_search(my\_list, element\_to\_find)

print(result)

Q.3) Write a program to display following pattern.

**1**

**2 3**

**4 5 6**

**7 8 9 10**

num=0

for i in range(1,6):

print()

for j in range(1,i+1):

num=num+1

print(num,end=" ")

**Q.4) singly linked list with create,insert and delete operations**

class Node:

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

self.data = data

self.next = None

class LinkedList:

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

self.head = None

def append(self, data):

new\_node = Node(data)

if not self.head:

self.head = new\_node

return

last\_node = self.head

while last\_node.next:

last\_node = last\_node.next

last\_node.next = new\_node

def display(self):

current\_node = self.head

while current\_node:

print(current\_node.data, end=" -> ")

current\_node = current\_node.next

print("None")

# Example usage:

my\_list = LinkedList()

# Create

my\_list.append(1)

my\_list.append(2)

my\_list.append(3)

# Display

print("Linked List after creation:")

my\_list.display()

# Display after insertion

print("\nLinked List after insertion:")

my\_list.display()

**Q5 Write a Python program to create a list of tuples with the first element as the number**

**and second element as the square of the number.**

lrange=int(input("enter the low range"))

urange=int(input("enter the upper range"))

a=[(x,x\*\*2)for x in range(lrange,urange+1)]

print(a)

**Q6 Implement searching algorithms to search an element using: Binary Search**

def bs(list1,n):

low=0

high=len(list1)-1

mid=0

while low<=high:

mid=(high+low)//2

if list1[mid]<n:

low=mid+1

elif list1[mid]>n:

high=mid-1

else:

return mid

return -1

list1=[10,7,8,6]

n=8

result=bs(list1,n)

if result!=-1:

print(n,"element found")

else:

print(n,"element not found")

**Q7 Write a python program to find repeated items in tuple**

t = (2,34,45,6,7,2,4,5,78,34,2)

**print**(t)

count = t.count(2

**print**(count)

**Q8 write singly list program to create,insert,delete display**

class Node:

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

self.data = data

self.next = None

class SinglyLinkedList:

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

self.head = None

def append(self, data):

new\_node = Node(data)

if not self.head:

self.head = new\_node

return

last\_node = self.head

while last\_node.next:

last\_node = last\_node.next

last\_node.next = new\_node

def delete\_at\_position(self, position):

if not self.head:

print("List is empty. Nothing to delete.")

return

if position == 0:

self.head = self.head.next

return

current\_node = self.head

for \_ in range(position - 1):

if current\_node is None or current\_node.next is None:

print("Position out of bounds.")

return

current\_node = current\_node.next

current\_node.next = current\_node.next.next

def display(self):

current\_node = self.head

while current\_node:

print(current\_node.data, end=" -> ")

current\_node = current\_node.next

print("None")

# Example usage:

my\_list = SinglyLinkedList()

# Create

my\_list.append(1)

my\_list.append(2)

my\_list.append(3)

# Display

print("Linked List after creation:")

my\_list.display()

# Delete at position

my\_list.delete\_at\_position(2)

print("\nLinked List after deletion:")

my\_list.display()

Q9 Doubly linked list create insert delete

class Node:

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

self.data = data

self.next = None

self.prev = None

class DoublyLinkedList:

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

self.head = None

def append(self, data):

new\_node = Node(data)

if not self.head:

self.head = new\_node

else:

last\_node = self.head

while last\_node.next:

last\_node = last\_node.next

last\_node.next = new\_node

new\_node.prev = last\_node

def print\_list(self):

current\_node = self.head

while current\_node:

print(current\_node.data, end=" <-> ")

current\_node = current\_node.next

print("None")

# Example Usage:

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

doubly\_linked\_list = DoublyLinkedList()

# Appending elements to the doubly linked list

doubly\_linked\_list.append(1)

doubly\_linked\_list.append(2)

doubly\_linked\_list.append(3)

doubly\_linked\_list.append(4)

# Printing the doubly linked list

doubly\_linked\_list.print\_list()

Q10 **Write a python program to add and remove operation on set**

s1={1,2,3,4}

s1.add(16)

print(s1)

s1.pop()

print(s1)

s1.remove(16)

print(s1)

Q11 **Sorting algorithm using quick sort**

def quick(arr):

if len(arr) <= 1:

return arr

else:

pivot = arr[0]

less = [x for x in arr[1:] if x <= pivot]

greater = [x for x in arr[1:] if x > pivot]

return quick(less) + [pivot] + quick(greater)

list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]

sorted\_list = quick(list)

print("Original List:", list)

print("Sorted List:", sorted\_list)

**Q12 Write a Python program to combine two dictionary adding values for common keys.**

**Sample Dictionary: d1={'a':100,'b':200,'c':300}**

**d2={'a':300,'b':200,'d':400}**

**Sample output: Counter({'a': 400, 'b': 400, 'd': 400, 'c':**

d1={'a':100,'b':200,'c':300}

d2={'a':100,'b':200,'c':300}

for key in d2:

if key in d1:

d2[key]=d2[key]+d1[key]

else:

pass

print(d2)

Q13 **Implement the sorting algoritm using bubble so**rt

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]

my\_list = [64, 34, 25, 12, 22, 11, 90]

print("Original List:", my\_list)

bubble\_sort(my\_list)

print("Sorted List:", my\_list)

**Q14 Write a Python program to create a dictionary from a string.**

**Sample-String:’W3resource’**

**Expected output: {'3': 1, 's': 1, 'r': 2, 'u'**

string='w3resource'

result\_dict={}

for char in string:

if char in result\_dict:

result\_dict[char]+=1

else:

result\_dict[char]=1

print(result\_dict)

Q15 **Implement sorting algorithm using merge sort**

def mergeSort(arr):

if len(arr) > 1:

a = len(arr)//2

l = arr[:a]

r = arr[a:]

mergeSort(l)

mergeSort(r)

b = c = d = 0

while b < len(l) and c < len(r):

if l[b] < r[c]:

arr[d] = l[b]

b += 1

else:

arr[d] = r[c]

c += 1

d += 1

while b < len(l):

arr[d] = l[b]

b += 1

d += 1

while c < len(r):

arr[d] = r[c]

c += 1

d += 1

def printList(arr):

for i in range(len(arr)):

print(arr[i], end=" ")

print()

if \_\_name\_\_ == '\_\_main\_\_':

arr = [0,1,3,5,7,9,2,4,6,8]

mergeSort(arr)

print("Sorted array is: ")

printList(arr)

**Q16 Write a python programto create an array of 5 integers and display the array and access them individually**

import array

a=array.array('i',[1,2,3,4,5])

print(a)

print(a[0])

print(a[1])

print(a[2])

print(a[3])

print(a[4])

Q 17 Implementing sorting algorithm using insertion sort

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

my\_list = [64, 34, 25, 12, 22, 11, 90]

print("Original List:", my\_list)

insertion\_sort(my\_list)

print("Sorted List:", my\_list)

**Q18 Write the python program to reverse the order of items in the array**

import array

a=array.array('i',[1,2,3,4])

print(a)

a.reverse()

print(a)

**Q19 Implement static stack operation init,isfull,isempty,push,pop,display**

class StaticStack:

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

self.capacity = capacity

self.stack = [None] \* capacity

self.top = -1

def is\_empty(self):

return self.top == -1

def is\_full(self):

return self.top == self.capacity - 1

def push(self, item):

if self.is\_full():

print("Stack Overflow: Cannot push element, the stack is full.")

else:

self.top += 1

self.stack[self.top] = item

print(f"Pushed {item} onto the stack.")

def pop(self):

if self.is\_empty():

print("Stack Underflow: Cannot pop element, the stack is empty.")

return None

else:

popped\_item = self.stack[self.top]

self.top -= 1

print(f"Popped {popped\_item} from the stack.")

return popped\_item

def display(self):

if self.is\_empty():

print("Stack is empty.")

else:

print("Stack elements:")

for i in range(self.top, -1, -1):

print(self.stack[i])

# Example usage:

stack\_capacity = 5

my\_stack = StaticStack(stack\_capacity)

my\_stack.push(1)

my\_stack.push(2)

my\_stack.push(3)

my\_stack.display()

popped\_item = my\_stack.pop()

if popped\_item is not None:

print(f"Popped item: {popped\_item}")

my\_stack.display()

**Q20 Implement static linear queue init,enqueue,deque,isempty,isfull,display**

class StaticQueue:

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

self.capacity = capacity

self.queue = [None] \* capacity

self.front = self.rear = -1

def is\_empty(self):

return self.front == self.rear == -1

def is\_full(self):

return (self.rear + 1) % self.capacity == self.front

def enqueue(self, item):

if self.is\_full():

print("Queue Overflow: Cannot enqueue element, the queue is full.")

else:

if self.is\_empty():

self.front = self.rear = 0

else:

self.rear = (self.rear + 1) % self.capacity

self.queue[self.rear] = item

print(f"Enqueued {item} into the queue.")

def dequeue(self):

if self.is\_empty():

print("Queue Underflow: Cannot dequeue element, the queue is empty.")

return None

else:

removed\_item = self.queue[self.front]

if self.front == self.rear:

self.front = self.rear = -1

else:

self.front = (self.front + 1) % self.capacity

print(f"Dequeued {removed\_item} from the queue.")

return removed\_item

def display(self):

if self.is\_empty():

print("Queue is empty.")

else:

print("Queue elements:")

i = self.front

while True:

print(self.queue[i], end=" ")

if i == self.rear:

break

i = (i + 1) % self.capacity

print()

# Example usage:

queue\_capacity = 5

my\_queue = StaticQueue(queue\_capacity)

my\_queue.enqueue(1)

my\_queue.enqueue(2)

my\_queue.enqueue(3)

my\_queue.display()

dequeued\_item = my\_queue.dequeue()

if dequeued\_item is not None:

print(f"Dequeued item: {dequeued\_item}")

my\_queue.display()

**Q21 Implementing dynamic stack using opeartion init,isfull,isempty,push,pop,display**

class Stack:

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

self.capacity = capacity

self.stack = []

self.top = -1

def is\_empty(self):

return self.top == -1

def is\_full(self):

return self.top == self.capacity - 1

def push(self, item):

if self.is\_full():

print("Stack Overflow: Cannot push element, the stack is full.")

else:

self.top += 1

self.stack.append(item)

print(f"Pushed {item} onto the stack.")

def pop(self):

if self.is\_empty():

print("Stack Underflow: Cannot pop element, the stack is empty.")

return None

else:

popped\_item = self.stack.pop()

self.top -= 1

print(f"Popped {popped\_item} from the stack.")

return popped\_item

def main():

capacity = int(input("Enter the capacity of the stack: "))

my\_stack = Stack(capacity)

while True:

print("\nStack Operations:")

print("1. Initialize Stack")

print("2. Check if Stack is Empty")

print("3. Check if Stack is Full")

print("4. Push Element onto Stack")

print("5. Pop Element from Stack")

print("6. Exit")

choice = input("Enter your choice (1-6): ")

if choice == "1":

print("Stack initialized.")

my\_stack = Stack(capacity)

elif choice == "2":

if my\_stack.is\_empty():

print("Stack is empty.")

else:

print("Stack is not empty.")

elif choice == "3":

if my\_stack.is\_full():

print("Stack is full.")

else:

print("Stack is not full.")

elif choice == "4":

item = input("Enter the element to push onto the stack: ")

my\_stack.push(item)

elif choice == "5":

popped\_item = my\_stack.pop()

if popped\_item is not None:

print(f"Popped item: {popped\_item}")

elif choice == "6":

print("Exiting the program.")

break

else:

print("Invalid choice. Please enter a valid option.")

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

main()

**Q22 Implementing dyanamic linear queue using operations init,is full,is empty,enqueue,dequeu,display**

class Queue:

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

self.capacity = capacity

self.queue = []

self.front = self.rear = -1

def is\_empty(self):

return self.front == self.rear == -1

def is\_full(self):

return (self.rear + 1) % self.capacity == self.front

def enqueue(self, item):

if self.is\_full():

print("Queue Overflow: Cannot enqueue element, the queue is full.")

else:

if self.is\_empty():

self.front = self.rear = 0

else:

self.rear = (self.rear + 1) % self.capacity

self.queue.append(item)

print(f"Enqueued {item} into the queue.")

def dequeue(self):

if self.is\_empty():

print("Queue Underflow: Cannot dequeue element, the queue is empty.")

return None

else:

removed\_item = self.queue.pop(0)

if self.front == self.rear:

self.front = self.rear = -1

else:

self.front = (self.front + 1) % self.capacity

print(f"Dequeued {removed\_item} from the queue.")

return removed\_item

def main():

capacity = int(input("Enter the capacity of the queue: "))

my\_queue = Queue(capacity)

while True:

print("\nQueue Operations:")

print("1. Initialize Queue")

print("2. Check if Queue is Empty")

print("3. Check if Queue is Full")

print("4. Enqueue Element into Queue")

print("5. Dequeue Element from Queue")

print("6. Exit")

choice = input("Enter your choice (1-6): ")

if choice == "1":

print("Queue initialized.")

my\_queue = Queue(capacity)

elif choice == "2":

if my\_queue.is\_empty():

print("Queue is empty.")

else:

print("Queue is not empty.")

elif choice == "3":

if my\_queue.is\_full():

print("Queue is full.")

else:

print("Queue is not full.")

elif choice == "4":

item = input("Enter the element to enqueue into the queue: ")

my\_queue.enqueue(item)

elif choice == "5":

dequeued\_item = my\_queue.dequeue()

if dequeued\_item is not None:

print(f"Dequeued item: {dequeued\_item}")

elif choice == "6":

print("Exiting the program.")

break

else:

print("Invalid choice. Please enter a valid option.")

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

main()

**Q23 Write a python function to sum of all the elements in a list**

def sum(n):

total=0

for x in n:

total+=x

return total

print(sum((8,2,3,4,5)))

**Q24 Write a python function to calculate the factorial of a number.the function accept the**

**number as an argument.**

def fact(n):

return 1 if(n==0 or n==1) else n\*fact(n-1)

num=int(input("enter a no"))

print("factorial of",num,"is",fact(num))

**Q25 Write a python function to check whether a number falls within a given range**

def func(n):

if n in range(3,9):

print(n,"falls within the range")

else:

print(n,"doesnt fall")

func(5)

**Q26 Write a python function that takes a list and returns a new list with distinct elements**

**from the first list**

def list(n):

x=[]

for a in n:

if a not in x:

x.append(a)

return x

print(list([1,1,2,2,4,8,9,9]))

**Q27 Write a program which accepts 6 integer values and prints “DUPLICATES” if any of**

**the values entered are duplicates otherwise it prints “ALL UNIQUE”**

print('Enter 6 numbers..')

a=list()

for i in range(6):

a.append(int(input('Enter: ')))

if len(set(a))!=len(a):

print('DUPLICATES.')

else:

print('Unique.')

**Q28 Write a python program to get the number of occurrences of specified elements in an**

**array.**

import array

a=array.array('i',[1,1,2,2,3,4,5])

print(a)

print("the occurences of 5 is",a.count(5))

**Q29 Write a Python script to generate and print a dictionary that contains a number**

**(Between 1 and n) in the form (x, x\*x)**

n=int(input("enter a no"))

d=dict()

for x in range(1,n+1):

d[x]=x\*x

print(d)

**Q30 Write a Python program to find maximum and the minimum value in a set**

S1={1,23,45,7}

print(max(s1))

print(min(s1))

**Q31 Write python program to do iteration over sets**

my\_set = {1, 2, 3, 4, 5}

print("Iterating over the set:")

for element in my\_set

: print(element)

**Q32 Write a Python program to check whether an element exists within a tuple.**

# t1=(1,2,3,4)

Print(3 in t1)

Print(“r” in t1)

**Q33 Write a Python program to get the 5th element from front and 5th element from last**

**of a tuple.**

T1=(1,2,3,4,5,6,7)

X=t1[5]

Y=t1[-5]

Print(x)

Print(y)

**Q34 Copy element 44 and 55 from the following tuple into a new tuple.**

**tuple1 = (11, 22, 33, 44, 55, 66)**

T1=(11,22,33,44,55,66)

T2=t1[3:5]

Print(t2)

**Q35 Reverse the following tuple ,Tup = (10, 20, 30, 40, 50)**

T=(10,20,30,40,50)

T1=reversed(t)

T=tuple(reversed(t))

Print(t)

Print(t1)