CHAPTER - 3
LISTS MANIPULATION & IMPLEMENTATION

Question 1:
Define a data structure.

Answer:
A data structure is a group of data which can be processed as a single unit. This group of data may be of similar or dissimilar data types. Data Structures are very useful while programming because they allow processing of the entire group of data as a single unit.

Question 2:
Name the two types of data structures and give difference between them.

Answer:
Data structures are of two types: Linear and Non – Linear.

1. In a linear data structure, the elements are stored in a sequential order.
2. In a non linear data structure, no sequential order is followed.
3. Linear Data Structure Examples: Arrays: Non-Linear Data Structure Examples: Tree, graph lists, stacks, queues, linked lists.

Question 3:
Give difference between an array and a list in Python.

Answer:
An array is defined as a set of contiguous data of similar data type. Python lists are actually arrays of variable length. The elements of a list are of eterogeneous types which means they are of different data types.

Question 4:
How are lists implemented in memory?
(or)
How is memory allocated to a list in Python?

Answer:
A list in Python is an array that contains elements (pointers to objects) of a specific size only and this is a common feature of all dynamically typed languages. For implementation of a list, a contiguous array of references to other objects is used. Python keeps a pointer to this array and the array's length is stored in a list head

structure. This makes indexing of a list independent of the size of the list or the value of the index. When items are appended or inserted, the array of references is resized.

Question 5:
What is sequential allocation of memory? Why do we say that lists are stored sequentially?

Answer:
A list is a allocated memory in sequential manner. This means that the elements of the list are stored in memory in sequence of their declaration. So, if you want to view the fifth element of the list, you have to first traverse through first four elements of the list. This is called sequential allocation of memory.

TOPIC-2
Searching Lists
Short Answer Type Questions-I (2 marks)

Question 1:
How is linear search different from binary search?

Answer:

1. Binary search requires the input data to be sorted near search doesn't.
2. Binary search requires an ordering comparison; linear search only requires equality comparisons
3. Binary search has complexity 0(log n); linear search has complexity O(n) as discussed earlier.
4. Binary search requires random access to the data; linear search only requires sequential access (this can be very important – it means a linear search can stream data of arbitrary size).

Short Answer Type Questions-II (2 marks)

Question 1:
Accept a list containing integers randomly. Accept any number and display the position at which the number is found in the list.

Answer:
```
maxrange = input("Enter Count of numbers: '')
marks=[]
flag=False
```

```
for i in range(0, maxrange):
marks. append(input(“ ?”))
number = inputfEnter number to be searched”)
for i in range(0, maxrange):
if marks [i]==number:
print number,“found at position”,i
flag=True
if flag==False:
print number, “not found in list”
```

Question 2:

What will be the status of the following list after the First, Second and Third pass of the slection sort method used for arranging the following elements in descending order?

Note : Show the status of all the elements after each pass very clearly underlining the changes.

12,14, -54,64,90,24

Answer:

	12	14	–54	64	90	24
Pass 1	90	**14**	–54	64	12	24
Pass 2	90	**64**	–54	14	12	24
Pass 3	90	**64**	24	14	12	–54

Question 3:

For a given list of values in descending order, write a method in Python to search for a value with the help of Binary search method. The method should return position of the value and should return -1 if the value not present in the list.

Answer:

```
def binarysrch (nums, x):
high=len (nums)
low=0
while low < high:
mid=(low+high)/2
midval=nums [mid]
if midval > x:
low = mid + 1
elif midval < x:
high = mid
else:
return mid
return -1
```

TOPIC-3
List Sorting
Short Answer Type Questions-I (2 marks)

Question 1:
What is bubble sort?

Answer:
Bubble sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements “bubble” to the top of the list. Because it only uses comparisons to operate on elements, it is also Called a comparison sort.

Question 2:
Write an algorithm for selection sort.

Answer:
Selection sort performs the following steps:

1. Starting at index 0, search the entire array to find the next smallest or largest value.
2. Swap the smallest or largest value found with the value at index 0.
3. Repeat steps 1 & 2 starting from the next index.

Question 3:
Explain insertion sort.

Answer:
Every repetition of insertion sort removes an element from the input data, inserting it into the correct position in the already-sorted list, until no input elements remain. The choice of which element to remove from the input is arbitrary and can be made using almost any choice algorithm.

Short Answer Type Questions-II (2 marks)

Question 1:
Write a function that takes a list that is sorted in ascending order and a number as argument. The function should do the following:

- Insert the number passed as argument in a sorted list.

- Delete the number from the list.

Answer:

```
from bisect import bisect
def listfunc(sortedlist,number) :
insert_point = bisect (sortedlist, number)
sortedlist.insert(insert_point,number)
print “List after Insertion”
print sortedlist
sortedlist.remove(number)
print “List after Deletion”
print sortedlist
maxrange = inputfEnter Count of numbers: ”)
numlist=[]
flag=False
for i in range(0, maxrange):
numlist.append(input(“ ?”))
numlist.sort()
number = inputfEnter the number”)
listfunc(numlist,number)
```

Question 2:
Consider the following list 95 79 19 43 52 3
Write the passes of bubble sort sorting the list in ascending order till the 3rd iteration.

Answer:
[79,19, 43, 52, 3, 95]-Pass 1
[19, 43,52,3,79, 95]-Pass 2
[19,43,3, 52, 79, 95]-Pass 3

Question 3:
Write a function that takes a sorted list and a number as an argument. Search for the number in the sorted list using binary search.

Answer:

```
def binary_search(SORTEDLIST, number):
low=0
high=len(SORTEDLIST)
found=False
while(low<high) and found==False:
mid=(int) (low+high/2)
if SORTEDLIST[mid]==number:
print “Number found at”,mid
found=True
break
```

```
elif SORTEDLIST[mid]<number:
low=mid+1
else:
high=mid-1
if low >= high:
print “Number not found”
maxrange = inputfEnter Count of numbers: ”)
numlist = []
for i in range(0, maxrange):
numlist.append(input(“?”))
numlist.sort()
print “Sorted list”,numlist
number = inputfEnter the number”)
binary_search(numlist,number)
```

Question 4:

In the following list containing integers, sort the list using Insertion sort algorithm. Also show the status of the list after each iteration.
15 -5 20 -10 10

Answer:

```
def insertion_sort(DATA_LIST):
for K in range (1, len(DATA_LIST)):
temp=DATA_LIST[K] # ptr=K-1,
while(ptr>=0) and DATA_LIST[ptr]>temp:
DATA_LIST[ptr+l]=DATA_LIST[ptr]
ptr=ptr-1
DATA_LIST [ptr+1] =temp
print DATA_LIST
DATA_LIST = [15,-5,20,-10,10]
print “LIST BEFOR SORTING”,DATAJ.IST
insertion_sort(DATA_LIST)
[-5,15, 20, -10,10]
[-5,15,20,-10,10]-Pass 1
[-10,-5,15,20,10]-Pass 2
[-10,-5,10,15, 20]-Pass 3
```

Question 5:

What will be the status of the following after the First, Second and Third pass of the insertion sort method used for arranging the following elements in descending order ?
Note: Show the status of all the elements after each pass very clearly underlining the changes.

Answer:

	22	24	–64	34	80	43
Pass 1	24	22	–64	34	80	43
Pass 2	24	22	–64	34	80	43
Pass 3	34	24	22	–64	80	43

Question 6:
Consider the following unsorted list:
99 78 25 48 51 11
Sort the list using selection sort algorithm. Show the status of the list after every iteration.

Answer:

```
def selection_sort(DATA_LIST):
for i in range(0, len (DATA_LIST)):
min = i
for j in range(i + 1, len(DATA_LIST)):
if DATA_LIST[j] < DATA_LIST[min]:
min = j
# swapping
temp= DATA_LIST[min]
DATA_LIST[min] = DATA_LIST[i]
DATA_LIST [i]=temp
print DATA_LIST
DATA_LIST = [99 78 25 48 51 11]
print “LIST BEFOR SORTING”,DATA_LIST
selection_sort(DATA_LIST)
```

Question 7:
Consider the following unsorted list Neena Meeta Geeta Reeta Seeta Sort the list using selection sort algorithm. Show the status of the list after every iteration.
(or)
Sort a list containing names of students in ascending order using selection sort.

Answer:

```
def selection_sort(DATA_LIST):
for i in range(0, len (DATA_LIST)):
min = i
for j in range(i + 1, len(DATA_LIST)):
if DATA_LIST[j] < DATA_LIST[min]:
min = j
# swapping
```

```
temp= DATA_LIST[min]
DATA_LIST[min] = DATA_LIST[i]
DATA_LIST[i]=temp
print DATA_LIST
NAME_LIST = [‘Neena’,’Beeta’,’Reeta’,’Geeta’,’Seeta’]
print “LIST BEFORE SORTING”,NAME_LIST
selection_sort(NAME_LIST)
LIST BEFORE SORTING [‘Neena’,‘Beeta’,‘Reeta’, ‘Geeta’, ‘Seeta’]
[‘Beeta’, ‘Neena’, ‘Reeta’, ‘Geeta’, ‘Seeta’]-Pass 1
[‘Beeta’, ‘Geeta’, ‘Reeta’, ‘Neena’, ‘Seeta’]-Pass 2
[‘Beeta’, ‘Geeta’, ‘Neena’, ‘Reeta’, ‘Seeta’]-Pass 3
[‘Beeta’, ‘Geeta’, ‘Neena’, ‘Reeta’, ‘Seeta’]-Pass 4
[‘Beeta’, ‘Geeta’, ‘Neena’, ‘Reeta’, ‘Seeta’]-Pass 5
```

Question 8:

A list contains rollno, name and marks of the student. Sort the list in descending order of marks using selection sort algorithm. [CBSE Text Book]

Answer:

```
def selection_sort(DATA_LIST):
for i in range(0, len (DATA_LIST)):
min = i
for j in range(i + 1, len(DATA_LIST)):
if DATA_LIST[j][2] > DATA_LIST[min][2]:
min = j
# swapping
DATA_LIST[min] [0] ,
DATA_LIST[i] [0] = DATA_LIST[i] [0] ,
DATA_LIST[mm] [0] DATA_LIST[min][l],
DATA_LIST[i][l] = DATA_LI ST [i] [ 1 ],
DATA_LI ST [min] [ 1 ] DATA_LIST[min] [2] ,
DATA_LIST[i] [2] = DATA_LIST[i] [2],
DATA_LIST[min] [2]
print DATA_LIST
maxrange=input(“Enter Number of Students: ”)
Students=[]
for i in range(maxrange):
Details=[]
Details. append(input(“Enter roll_no”))
Details.append(raw_input(“Enter name”))
Details. append(inprft(“Enter marks”))
Students.append(Details)
print Students
selection_sort(Students)
```

Question 9:
What will be the status of the following list after the First. Second and Third pass of the insertion sort method used for arranging the following elements in descending order ?
12,34,46, -34,90,23
Note : Show the status of all the elements after each pass very clearly underlining the changes.

Answer:

output of diferent passes Pass One
[12,34,46,-34,90,23], {}
Pass Two
[34, 46, -34, 90, 23} , {12}
Pass Three
LISTS MANIPULATION AND IMPLEMENTATION
[46, -34, 90, 23} , {12, 34} Pass Six
Pass Four {23}, {-34,12, 34, 46, 90}
[-34, 90, 23} , { 12, 34, 46} Pass Seven
Pass Five {} , {-34,12, 23, 34, 46, 90}
{90,23} , {-34,12,34,46} It is the sorted list.

Long Answer Type Questions (4 marks)

Question 1:
Consider the following unsorted list: 90 78 20 46 54 1 Write the list after:

1. 3rd iteration of selection sort
2. 4th iteration of bubble sort
3. 5th iteration of insertion sort Ans. Working :

Answer:

Working :

Bubble Sort	Selection Sort	Insertion Sort
[1, 90, 78, 46, 54, 20]	[1, 78, 20, 46, 54, 90]	[78, 90, 20, 46, 54, 1]
[1, 20, 90, 78, 54, 46]	[1, 20, 78, 46, 54, 90]	[20, 78, 90, 46, 54, 1]
[1, 20, 46, 90, 78, 54]	[1, 20, 46, 78, 54, 90]	[20, 46, 78, 90, 54, 1]
[1, 20, 46, 54, 90, 78]	[1, 20, 46, 54, 78, 90]	[20, 46, 54, 78, 90, 1]
[1, 20, 46, 54, 78, 90]	[1, 20, 46, 54, 78, 90]	[1, 20, 46, 54, 78, 90]
[1, 20, 46, 54, 78, 90]	[1, 20, 46, 54, 78, 90]	

1. 3rd iteration of selection sort: [1,20,46,78,54,90]
2. 4th iteration of bubble sort: [1, 20, 46, 54, 90, 78]
3. 5th iteration of insertion sort: [1, 20, 46, 54, 78, 90

Question 2:

Consider the following unsorted list:
10 5 55 13 3 49 36
Write the position of elements in the list after:
(i) 5th iteration of bubble sort
(ü) 7th iteration of insertion sort
(iii) 4th iteration of selection sort

Answer:

Working :

Bubble Sort	Selection Sort	Insertion Sort
[3, 10, 55, 13, 5, 49, 36]	[3, 5, 55, 13, 10, 49, 36]	[5, 10, 55, 13, 3, 49, 36]
[3, 5, 55, 13, 10, 49, 36]	[3, 5, 55, 13, 10, 49, 36]	[5, 10, 55, 13, 3, 49, 36]
[3, 5, 10, 55, 13, 49, 36]	[3, 5, 10, 13, 55, 49, 36]	[5, 10, 13, 55, 3, 49, 36]
[3, 5, 10, 13, 55, 49, 36]	[3, 5, 10, 13, 55, 49, 36]	[3, 5, 10, 13, 55, 49, 36]
[3, 5, 10, 13, 36, 55, 49]	[3, 5, 10, 13, 36, 49, 55]	[3, 5, 10, 13, 49, 55, 36]
[3, 5, 10, 13, 36, 49, 55]	[3, 5, 10, 13, 36, 49, 55]	[3, 5, 10, 13, 36, 49, 55]
[3, 5, 10, 13, 36, 49, 55]	[3, 5, 10, 13, 36, 49, 55]	

1. 5th iteration of bubble sort: [3, 5,10,13, 36, 55, 49]
2. Insertion sort doesn't have the 7th iteration
3. 4th iteration of selection sort: [3, 5,10,13, 55, 49, 36]

Question 3:

Write a class CITY in Python with following
– CName # String Value specifications :
– Pop # Numeric value for Population instance attributes
-KM # Numeric value

– Ccode # Numeric value
– Density #Numeric value for Population Density

Methods:

– Dencal() #Method to calculate Density as Pop/KM
– Record() #Method to allow user to enter values Ccode, CName, Pop, KM and call DenCal ( ) method
– View() #Method to display all the members also display a message “Highly Populated City” if the Density is more than 10000.

Answer:

```
class CITY:
def_init_(self): self.Ccode = 0
self.CName = self. Pop = 0
self.KM = 0 self.Density = 0
def DenCal (self) :
self.Density = self.Pop/self. KM
def Record (self)
self Ccode=input (“Enter Ccode”)
self.CName=raw_input (“Enter CName”)
self.Pop=input (“Enter population”)
self.KM=input (“Enter KM”)
DenCal(self) //or self.DenCal()
def View (self):
print Ccode, CName, Pop, KM, Density
if self.Density > 10000:
print (“Highly populated city”)
# OR print (“Highly populated city”)
```

Question 4:

A list contains Item_code, Item_name, qty and price. Sort the list :

- In ascending order of qty using Bubble sort.
- In descending order of price using Insertion sort.

Answer:

```
def bubble_sort(DATA_LIST) :
i = 0
j = 0
l = len(DATA_LIST)
for i in range(l):
print “Bubble Sort Iterations – Asc order of Quantity”
for j in range(i+l,l):
if DATA_LIST[i][3] > DATA_LIST[j][3]:
# swapping
```

```
DATA_LIST[i][0], DATA_LIST[j] [0]=DATA LIST[j] [0],DATA_LIST[i] [0]
DATA_LIST[i][1], DATA_LIST[j][1]=DATA_ LIST[j][l],DATA_LIST[i][l]
DATA_LIST[i] [2], DATA_LIST[j] [2]=DATA_ LIST[j][2],DATA_LIST[i][2]
DATA_LIST[i][3], DATA_LIST[j][3]=DATA_ LIST[j][3] ,DATA_LIST[i] [3]
print DATA_LIST
def insertion_sort(DATA_LIST):
for K in range (1, len(DATA_LIST)):
temp=DATA_LIST[K][2]
ptr=K-1
print “Insertion Sort Iterations – Desc order of price”
while(ptr>=0) and DATA_LIST[ptr][2] < temp:
DATA_LIST[ptr+1] [0]=DATA_LIST[ptr] [0]
DATAJHST [ptr+1] [1]=DATA_LIST[ptr] [1]
DATA_LIST[ptr+1][2]=DATA_LIST[ptr][2]
DATA_LIST[ptr+1] [3]=DATA_LIST[ptr] [3]
ptr=ptr-1
DATA_LIST[ptr+1][2]=temp
print DATA_LIST
maxrange = input(“Enter Number of Items: “)
Items=[]
for i in range (maxrange):
Details=[]
Details. append(input(“Enter Item Code”))
Details.append(raw_input(“Enter Item name”))
Details.append(float(raw_input(“Enter price”)))
Details.append(input(“Enter Quantity”)) Items. append(Details)
print “BEFORE SORTING”,Items bubble_sort(Items)
insertion_sort(Items)
```

TOPIC-4
Stacks And Queues With Lists
Very Short Answer Type Questions (1 mark)

Question 1:
Expand the following:

1. LIFO
2. FIFO

Answer:

1. LIFO: Last-In First-Out

2. FIFO: First-In First-Out

Question 2:
Define stack.

Answer:
A stack is a data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the top of the stack; the only element that can be removed is the element that was at the top of the stack.

Question 3:
Define Queue.

Answer:
Queue is a first-in, first-out (FIFO) data structure, i.e. the element added first to the queue will be the one to be removed first. Elements are always added to the rear of the queue and removed from the front of the queue.

Question 4:
Write all the operations possible in data structure.

Answer:
The major operations are:

1. Traversal
2. Insertion
3. Deletion
4. Searching

Question 5:
What are the two major queue operations?

Answer:
Addition of element is known as INSERT operation, also known as enqueuing. It is done using rear terminal position, i.e. tail end. Removal of element is known as DELETE operation, also known as dequeueing.

Short Answer Type Questions-I (2 marks)

Question 1:
Write an algorithm to implement push operation.

Answer:

1. Start
2. Initialize top with -1.
3. Input the new element.
4. Increment top by one.
5. stack[top]=new element
6. Print “Item Inserted”
7. Stop

Question 2:
Write an algorithm to implement pop operation.

Answer:

1. Start
2. If the value of top is -1 go to step 3 else go to step 4
3. Print “Stack Empty” and go to step 7
4. Deleted item = Stack[top]
5. Decrement top by 1
6. print “Item Deleted”
7. Stop

Question 3:
Write an algorithm to implement insertion opera¬tion of queue.

Answer:

1. Start
2. Check FRONT and REAR value,
    1. if both the values are -1, then FRONT and REAR are incremented by 1
    2. other wise Rear is incremented by one.
3. queue [Rear]=new element.
4. Stop

Question 4:
Write an algorithm to implement deletion operation of queue.

Answer:

1. Start
2. Check for underflow situation by checking value of Front=-1
    1. If it is display appropriate message and stop
    2. Otherwise Deleted item=queue [Front]
3. If Front=Rear then Front=Rear=-1 Otherwise Front is incremented by one

4. Print “Item Deleted”
5. Stop

Question 5:
For a given list of values in ascending order, write a method in Python to search for a value with the help of Binary Search method. The method should return position of the value and should return -1 if the value not present in the list.

Answer:
```
def Binary Search (list, search):
lower_bound=0
upper_bond=len(list) -1
found=false
pos=’x’
while lower_bound< =upper_bond:
middle_pos=(lower_bound + upper_bond)/2
if list [middle_pos] = = search:
pos=middlepos found=True break
elif search < list [middle_pos]:
upper_bound=middle_pos-l else:
lower_bound=middle_pos + 1 if found:
print(“The item found”, pos) else:
print (“The item not available”)
return (-1)
```

Long Answer Type Questions (4 marks)

Question 1:
Write Insert (city) and Delete (City) methods in Python to add City and Romave City considering them to act as Insert and Delete operations of the data structure Queue.

Answer:
```
class queue:
city=[ ]
def Insert (self):
a = raw_input(“Enter city”)
queue.city append (a)
def Delete (self):
if (queue, city = = []):
print “Queue empty”
else:
print “Deleted element is”,
queue.city[0]
```

```
queue.city.delete ()
OR
class queue:
city=[ ]
def Insert(self):
a=raw_input("Enter city")
queue.a.append(a)
def Delete(self):
if (queue.city==[ ]):
print ("Queue emtpy")
else
print ("Deleted element is",queue, city [0])
```

Question 2:
Define stack class in Python to operate on stack of numbers.

Answer:

```
class Stack:
def_init_(self):
self.s = [ ]
def push(self):
a = input("Enter number")
self.s. append(a)
def pop(self):
self.s.pop()
def display (self):
l = len(self.s)
for I in range(l-1,1,-1):
print self.s[i]
```

Question 3:
Write a function to push any student's information to stack.

Answer:

```
def push(stack):
s=[]
print "STACK BEFORE PUSH" display(stack)
s.append(input("Enter student rollno?"))
s.append(raw_input("Enter student name"))
s.append(raw_input("Enter student grade"))
stack, append(s)
def display (stack):
l=len(stack)
print "STACK CONTENTS"
for i in range(l-1,-1,-1):
print stack[i]
```

```
stack=[]
print “Creating Stack”
n = input(“Enter the number of students”)
for i in range(n): student = []
student.append(input(“Enter student rollno?”))
student.append(raw_input(“Enter student name”))
student.append(raw_input(“Enter student grade”))
stack, append(student) push(stack)
display(stack)
```

Question 4:
Write the push operation of stack containing names using class.

Answer:

```
class stack:
s=[]
def push(self):
a=r’aw_input(“Enter any name :”)
stack.s.append(a)
def display(self):
l=len(stack.s) for i in range(l-1,-1,-1):
print stack.s[i]
a=stackO
n= input(“Enter no. of names”)
for i in range (n):
a.push()
a.display()
```

Question 5:
Write Insert (Place) and Delete (Place) methods in Python to be add Place and Remove Place considering them to act as Insert and Delete operations of the data structure Queue.

Answer:

```
Class queue: place=[]
def Insert (self):
a=raw_input (“Enter city”)
queue.place. append (a)
def delete (self):
if (queue.place = = []):
print “Queue empty”
else:
print “Deleted element is”, queue, place [0]
class queue :
place = [ ]
def Insert (self) 1
```

```
a = rawjnput (“Enter place”) queue, a.append (a)
def delete (self):
if (queue.place==[ ] ):
print (“Queue empty”)
else:
print (“Deleted element is” queue, place [0])
queue.place.delete ( )
```

Question 6:
Write the pop operation of stack containing names using class.

Answer:

```
class stack:
s=[]
def push (self):
a=raw_input(“Enter any name :”)
stack.s.append(a)
def pop(self):
if (a.s==D):
print “Stack Empty”
else:
print “Deleted element is : ”,
a.s.pop()
def display (self):
l=len(stack.s)
print “STACK CONTENTS”
for i in range(l-1,-1,-1):
print stack.s[i]
a=stack()
n= input(“Enter no. of names”)
for i in range(n):
a.push()
a.pop()
a.display()
```

Question 7:
Write the pop operation of stack containing num-bers using class.

Answer:

```
class stack:
s=[]
def push(self):
a=input(“Enter number :”)
stack.s.append(a)
def pop(self):
if (a.s==[]):
```

```
print “Stack Empty”
else:
print “Deleted element is : ”,a.s.pop()
def display(self):
l=len(stack.s) print “STACK CONTENTS”
for i in range(l-1,-1,-1):
print stack.s[i]
a=stack()
n= input(“Enter count of numbers”)
for i in range (n):
a.push()
a.pop()
a. display ()
```

Question 8:
Write a function to add any customer’s information to queue.

Answer:

```
def insert(queue): customer=[]
print “QUEUE BEFORE INSERT” display(queue)
customer.append(input(“Enter customer number?’’))
customer.append(raw_input(“Enter customer name”))
customer.append(input(“Enter customer phone number”))
queue. append(customer)
def display (queue):
l=len(queue)
fori in range(O.l): print queue[i]
queue=[]
print “Creating Queue”
n=input(“Enter the number of customers”)
for i in range(n):
customer=[]
customer. append(input(“Enter customer number?”))
customer.append(rawjnput(“Enter customer name”))
customer.append(input(“Enter customer phone number”))
queue.append(customer)
insert(queue)
display(queue)
```

Question 9:
Write the insertion operation of queue containing character using class.

Answer:

```
class queue:
s=0
def insert(self):
```

```
a=raw_input("Enter any letter:")
queue.s.append(a)
def display (self):
l=len(queue.s)
print "QUEUE CONTENTS"
for i in range(o,l):
print queue. s[i]
a=queue()
n= input("Enter no. of letters")
for i in range(n): a.insert()
ch = raw_input("Do you want to insert more letters")
if ch=='y':
a.insert()
a.display()
```

Question 10:
Write Add(Fruit) and Remove (Fruit) methods in Python to insert name of a Fruit and to delete name of a Fruit considering them to act as Insert and Delete operations of the data structure Queue.

Answer:

```
def que insert (fruit, Rear) :
ch='y'
while ch=='y' or ch=='yes'
Frt=input("Enter a fruit")
Rear=Rear + 1
Fruit, append (Frt)
print("Do you wand to Add more")
ch=input ().upper ( )
if ch=='N' or ch=='No' :
break
return rear
def quedelete (Fruit, Rear):
if not len (fruist):
print f("Fruit + list is empty")
else :
Rear = Rear – 1
Frt = Fruit.pop( )
print("Fruit is delete from list")
return Rear
```

Question 11:
Write the deletion operation of queue containing characters using class.

Answer:

```
class queue:
```

```
s=0
def insert(self):
a=raw_input(“Enter any letter:”)
queue.s.append(a)
def delete(self):
if (a.s==[]):
print “Queue Empty”
else:
print “Deleted element is: ”,queue.s[0]
del queue.s[0]
def display(self): l=len(queue.s)
print “QUEUE CONTENTS”
for i in range(0,l):
print queue.s[i]
a=queue()
n= input(“Enter no. of letters”)
for i in range (n):
a.insert()
a. delete()
a. display ()
```

Question 12:
Write the deletion operation of queue containing numbers using class.

Answer:

```
class queue:
s=[]
def insert(self):
a=input(“Enter the number:”)
queue.s.append(a)
def delete(self):
if (a.s==Q):
print “Queue Empty”
else:
print “Deleted element is: ”,queue.s[0]
del queue.s[0]
def display(self):
l=len(queue.s)
print “QUEUE CONTENTS”
for i in range(0,l):
print queue.s[i]
a=queue()
n= input(“Enter no. of numbers”)
for i in range(n):
a.insert()
```

a.delete()
a.display()

TOPIC-5
Applications of Stacks
Very Short Answer Type Questions (1 mark)

Question 1:
Give one example of infix expression.

Answer:
A + B *C / D

Question 2:
Give one example of postfix expression.

Answer:
ABC*D/+

Question 3:
Give one example of prefix expression.

Answer:
+/*BCDA

Question 4:
Convert (A+B)*C in to postfix form.

Answer:
AB+C*

Short Answer Type Questions-I (2 marks)

Question 1:
Evaluate the following postfix notation of expressio,
Show status of stack after every operation.
12,2,7,34,20,-, +,5, +

Answer:

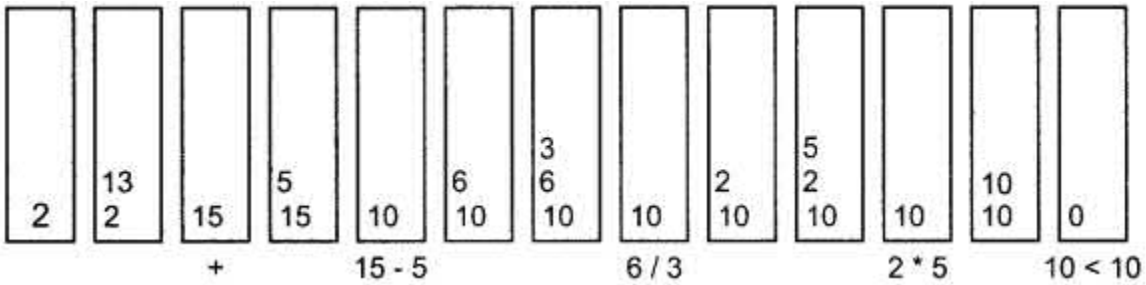
Element	Stack
12	12

2	12, 2
/	6
34	6, 34
20	6, 34, 20
–	6, 14
+	20

5	20, 5
+	25

Question 2:
Evaluate the following postfix expression. Show the status of stack after execution of each operation separately: 2,13, + , 5, -,6,3,/,5,*,<

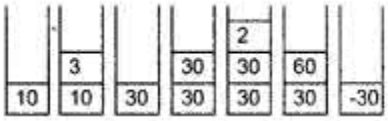
Answer:
Stack contents



Result: 0

Question 3:
Evaluate using stack 10, 3,*, 30, 2,*,-

Answer:



Question 4:
Evaluate the following Postfix expression :
20,10,-15,3,/, +, 5, *

Answer:

Symbol	Operation	Stack	Result
20	Push	20	
10	Push	20, 10	
-	Pop (10) Pop (20) Push (20-10) = 10	10	
15	Push	10, 15	
3	Push	10, 15, 3	
/	Pop (3) Pop (15) Push (15/3)=5	10.5	
+	Pop (5) Pop (10) Push (10+5) = 15	15	
5	Push	15,5	
*	Pop (5) Pop (15) Push (15*5) = 5	75	75

Question 5:
Convert A+ (B*C – (D / E A F) * G) * H into postfix form showing stack status after every step.

Answer:

S. No.	Element	Stack	Expression
1	A	(	A
2	+	(+	A
3	(	(+(	A
4	B	(+(	AB
5	*	(+(*	AB
6	C	(+(*	ABC
7	-	(+(-	ABC*
8	(	(+(-(	ABC*
9	D	(+(-(	ABC*D
10	/	(+(-(\	ABC*D
11	E	(+(-(\	ABC*DE
12	^	(+(-(\^	ABC*DE
13	F	(+(-(\^	ABC*DEF
14	)	(+(-	ABC*DEF^/
15	*	(+(-*	ABC*DEF^/
16	G	(+(-*	ABC*DEF^/G
17	)	(+	ABC*DEF^/G*-
18	*	(+*	ABC*DEF^/G*-
19	H	(+*	ABC*DEF^/ G *- H
20	)		ABC*DEF^/ G *- H*+

Question 6:

Convert (True And False) OR (False And True) into postfix form using stack.

Answer:

S. No.	Element	Stack	Expression
1	(	((	True
2	True	((	True
3	And	((And	True
4	False	((And	True False
5	)	(	True False And
6	OR	(OR	True False And
7	(	(OR (	True False And
8	False	(OR (	True False And False
9	And	(OR (And	True False And False
10	True	(OR (And	True False And False True
11	)	(OR	True False And False True And
12	)		True False And False True And OR

Question 7:
What are the applications of queue in computer?

Answer:
In a single processor multitasking computer, job(s) wait to be processed form a queue. Same happens when we share a printer with many computers.

- Compiling a HLL code.
- Using download manager, for multiple files also uses queue for ordering the files.
- In multiuser OS – job scheduling is done through queue.

Question 8:
Evaluate the following postfix notation of expression. Show status of stack after every operation.

Answer:

Element	Stack
22	22
11	22, 11
/	2
14	2, 14
10	2, 14, 10
–	2, 4
+	6
5	6, 5
–	1

Final Result = 1

Question 9:
Evaluatethefollowingpostixnotationofexpression. Show status of stack after every operation.
12,2, *, 24,20, -, +, 8, –

Answer:

Scanned Element	Stack Status	Expression
12	1	–
2	12,	–
*	12, 2, *	24
24	24, 24	24
20	24, 24, 20	24
–	24, 24, 20 –	4
+	24, 4 +	4
8	28, 8	4
–	28, 8, –	20
Nill	20	20

Question 10:
Give few applications of stack.

Answer:

1. Expression evaluation
2. Backtracking (game playing, finding paths, exhaustive searching).
3. Memory management, run-time environment for nested language features.

Short Answer Type Questions-II (2 marks)

Question 1:
Write an algorithm to evaluate postfix expression.

Answer:
Step 1: Start
Step 2: Check all symbols from left to right and repeat steps 3 & 4 for each symbol of expression 'E' until all symbols are over.

1. If the symbol is an operand, push it onto stack.
2. If the symbol is an operator then:
3. Pop the top two operands from stack and apply an operator in between them.
4. Evaluate the expression and place the result back on the stack.

Step 3: Set result equal to top element on the stack.
Step 4: Stop

Question 2:
Write an algorithm to convert infix to postfix.

Answer:

1. Create an empty stack and an empty postfix output string/stream.
2. Scan the infix input string/stream left to right.
3. If the current input token is an operand, simply append it to the output string (note the examples above that the operands remain in the same order).
4. If the current input token is an operator, pop off all operators that have equal or higher precedence and append them to the output string; push the operator onto the stack. The order of popping is the order in the output.
5. If the current input token is push it onto the stack.
6. If the current input token is ')', pop off all operators and append them to the output string until a '(' is popped; discard the '('.
7. If the end of the input string is found, pop all operators and append them to the output string.