Dr. D. Y. Patil Pratishthan's
D. Y. Patil Institute of Master of Computer Applications & Management
(Approved by AICTE, Recognized by DTE, Mah.; Affiliated to SPPU)

DSA Part A: Preliminary Term Journal Submission

MCA – I (Sem – I) **Academic Year 2024-2025 (Term: Aug - Dec 2024)**
Lab: Data Structure and Algorithm **CO Mapped=CO4 BT Level 3=Apply and 6=Create**

SR NO	Programs	Date	Sign
1	Create Array and perform the following operations: a)Update array b)slice array element c)extend array d)append array e)remove element.	19/9/24	
2	Create an empty list, take size of list from user and append oneone element in empty list.	19/9/24	
3	Create a list and perform following operations: a)update b)count c)sort d)insert e)append f)extend g)pop	1/10/24	
4	Create function Employee details with user define data like empid, empname, empqualification, empdesignation, empsalary, emp_currentmonthrank. If current month rank is >4 than add incentive Rs.2500 in salary.	1/10/24	
5	Create class Pizza_order_management_system with the class attributes like ordered, Pizza_order, Pizza_type, Quantity, Address, Distance_Kilometer. If the customer distance kilometer is greater than 3.5 than Rs 45 will the extra charge will be added in bill.Print bill details in printBill()	1/10/24	
6	Create Singly linked list and perform the following operations: a)Insert from start b)Insert from end c)traverse	3/10/24	
7	Create Singly linked list and perform the following operations: a)Insert from specific position b)traverse	3/10/24	

Program 1:

Create Array and perform the following operations: a)Update array b)slice array element c)extend array d)append array e)remove element.

Solution:

```
import array as arr
data = arr.array('i',[10,20,30,40,50])
print(data)
print("Array after updating: ")
data[2] = 90
print(data)
print("After slicing array from 2nd position to 3rd position: ")
print(data[1:4])
print("After extending array: ")
data.extend([60,70,80])
print(data)
print("After appending array: ")
data.append(100)
print(data)
print("After removing element: ")
data.remove(40)
print(data)
```

Output:

```
array('i', [10, 20, 30, 40, 50])
Array after updating:
array('i', [10, 20, 90, 40, 50])
After slicing array from 2nd position to 3rd pos
array('i', [20, 90, 40])
After extending array:
array('i', [10, 20, 90, 40, 50, 60, 70, 80])
After appending array:
array('i', [10, 20, 90, 40, 50, 60, 70, 80, 100]
After removing element:
array('i', [10, 20, 90, 50, 60, 70, 80, 100])
```

Program 2:

Create an empty list, take size of list from user and append oneone element in empty list.

Solution:

```
data=[]
n = int(input("Enter the size of array: "))
for i in range(0,n):
    data.append(int(input(f"Enter the element at {i} position: ")))
print(data)
```

Output:

```
Enter the size of array:  3
Enter the element at 0 position:  10
Enter the element at 1 position:  20
Enter the element at 2 position:  30
[10, 20, 30]
```

Program 3:

Create a list and perform following operations: a)update b)count c)sort d)insert e)append f)extend g)pop

Solution:

```
data = [34,5,67,43,5,87,89]
print(data)
print("list after updation: ")
data[2]=90
print(data)
print("The element 5 is occured",data.count(5),"Times")
print("list after sorting: ")
data.sort()
print(data)
print("Inserting element 45 at position 4")
data.insert(3,45)
print(data)
print("After appending the list")
data.append(100)
print(data)
print("After extending the list with 3 elements")
data.extend([4,5,6])
print(data)
print("After popping last element")
data.pop()
print(data)
```

Output:

```
[34, 5, 67, 43, 5, 87, 89]
list after updation:
[34, 5, 90, 43, 5, 87, 89]
The element 5 is occured 2 Times
list after sorting:
[5, 5, 34, 43, 87, 89, 90]
Inserting element 45 at position 4
[5, 5, 34, 45, 43, 87, 89, 90]
After appending the list
[5, 5, 34, 45, 43, 87, 89, 90, 100]
After extending the list with 3 elements
[5, 5, 34, 45, 43, 87, 89, 90, 100, 4, 5, 6]
After popping last element
[5, 5, 34, 45, 43, 87, 89, 90, 100, 4, 5]
```

Program 4:

Create function Employee details with user define data like empid, empname, empqualification, empdesignation, empsalary, emp_currentmonthrank. If current month rank is >4 than add incentive Rs.2500 in salary.

Solution:

```
def emp_details():
    emp_id=int(input("Enter the employee id: "))
    emp_name=(input("Enter the employee name: "))
    emp_qualification=(input("Enter the employee qualification: "))
    emp_designation=(input("Enter the employee designation: "))
    emp_salary=int(input("Enter the employee salary: "))
    emp_currentmonthrank=int(input("Enter the employee's current month rank: "))
    print("----------------------------------------------------------------")
    print("Id=",emp_id)
    print("Name=",emp_name)
    print("Qualification=",emp_qualification)
    print("Designation=",emp_designation)
    print("Salary=",emp_salary)
    print("Current month rank=",emp_currentmonthrank)
    if emp_currentmonthrank>4:
        emp_salary=emp_salary+2500
        print(f"New salary of employee {emp_name} is :",emp_salary)
    else:
        print("Salary=",emp_salary)
emp_details()
```

Output:

```
Enter the employee id:  100
Enter the employee name:  Sanket
Enter the employee qualification:  MCA
Enter the employee designation:  Senior Software Engineer
Enter the employee salary:  100000
Enter the employee's current month rank:  5
Id= 100
Name= Sanket
Qualification= MCA
Designation= Senior Software Engineer
Salary= 100000
Current month rank= 5
New salary of employee Sanket is : 102500
```

Program 5:

Create class Pizza_order_management_system with the class attributes like ordered, Pizza_order, Pizza_type, Quantity, Address, Distance_Kilometer. If the customer distance kilometer is greater than 3.5 than Rs 45 will the extra charge will be added in bill.Print bill details in printBill()

Solution:

```
class Pizza_order_management_system():
    def __init__(self):
        self.cust_Name=input("Enter customer name: ")
        self.cust_MoNo=input("Enter customer Mo No:")
        self.pizza_Name=input("Enter Name of pizza: ")
        self.pizza_Type=input("Enter pizza type -> 1)Small 2)Extra 3)Large 4)Extra Large\n")
        self.pizza_Price=int(input("Enter Price of pizza: "))
        self.Quantity=int(input("Enter quantity: "))
        self.Address=input("Enter address for delivery: ")
        self.distance_Km=int(input("Enter the distance in Kilometer: "))

    def printBill(self):
        self.Total=self.pizza_Price*self.Quantity
        if self.distance_Km>3.5:
            self.dist=self.distance_Km-3.5
            self.Total=self.Total+45*round(self.dist)
        print("\n\n\n")
        print("-----------------------------------------")
        print(self.cust_Name)
        print(self.cust_MoNo)
        print("Order_>  ",self.pizza_Name,"       ",self.pizza_Type)
        print("Total-> ",self.pizza_Price," *  ",self.Quantity,"   =   ",self.Total)
        print("Address: ",self.Address)
        print("-----------------------------------------")

pizza=Pizza_order_management_system()
pizza.printBill()
```

Output:

```
Enter customer name:  Jonhy English
Enter customer Mo No: 1234567890
Enter Name of pizza:  Veg Pizza
Enter pizza type -> 1)Small 2)Extra 3)Large 4)Extra Large
 Small
Enter Price of pizza:  200
Enter quantity:  4
Enter address for delivery:  Akurdi,411035
Enter the distance in Kilometer:  5



-------------------------------------------
Jonhy English
1234567890
Order_>   Veg Pizza          Small
Total->  200  *   4    =    890
Address:  Akurdi,411035
-------------------------------------------
```

Program 6:

Create Singly linked list and perform the following operations: a)Insert from start b)Insert from end c)traverse

Solution:

```
class Node():
    def __init__(self,data):    #Create a new Node
        self.data=data
        self.next=None
class LinkedList():
    def __init__(self):         #Define head and tail
        self.head=None
        self.tail=None
    def insert_beg(self,data):        #Insert at beginning
        newNode=Node(data)
        if self.head==None:
            self.head=self.tail=newNode
        else:
            newNode.next=self.head
            self.head=newNode
    def insert_end(self,data):          #Insert at End
        newNode=Node(data)
        if self.head==None:
            self.head=self.tail=newNode
        else:
            self.tail.next=newNode
            self.tail=newNode
    def display(self):            #Traverse the linked list
        traverse=self.head
        while traverse!=None:
            print(traverse.data,end="-->")
            traverse=traverse.next
        print("None")

ll=LinkedList()
```

```
ll.insert_beg(10)
ll.insert_beg(20)
ll.insert_end(30)
ll.insert_end(40)
ll.display()
```

Output:

```
20-->10-->30-->40-->None
```

Program 7:

Create Singly linked list and perform the following operations: a)Insert from specific position b)traverse

Solution:

```
class Node():
    def __init__(self,data):    #Create a new Node
        self.data=data
        self.next=None
class LinkedList():
    def __init__(self):         #Define head and tail
        self.head=None
        self.tail=None
    def insert_beg(self,data):        #Insert at beginning
        newNode=Node(data)
        if self.head==None:
            self.head=self.tail=newNode
        else:
            newNode.next=self.head
            self.head=newNode

    def insert_spec_loc(self,data,loc):        #Insert at specific position
        cNode=self.head
        cLoc=1
        newNode=Node(data)
        if self.head==None:
            self.head=self.tail=newNode
        elif loc==1:
            newNode.next=self.head
            self.head=newNode
        else:
            while cNode.next!=None and cLoc<loc-1:
                cNode=cNode.next
                cLoc=cLoc+1
            newNode.next=cNode.next
```

```
            cNode.next=newNode

    def display(self):          #Traverse the linked list
        traverse=self.head
        while traverse!=None:
            print(traverse.data,end="-->")
            traverse=traverse.next
        print("None")

ll=LinkedList()
ll.insert_beg(30)
ll.insert_beg(20)
ll.insert_beg(10)
ll.display()
print("After inserting 100 at position 2nd")
ll.insert_spec_loc(100,2)
ll.display()
```

Output:

```
10-->20-->30-->None
After inserting 100 at position 2nd
10-->100-->20-->30-->None
```