HARSH KASLIWAL
2003085
C21

ASSIGNMENT 1

1. Python program to implement operations of linked list
    1. Display list
    2. Insert at beginning
    3. Insert at End
    4. Insert at specified postion
    5. Delete from beginning
    6. Delete from end
    7. Delete at specified postion
    8. Delete a particular element
    9. Search an element
    10. Replace element at specified index
    11. Forward traversal
    12. Reverse traversal
    13. Insert after an element

CODE:

```
class Node:     def

__init__(self,value,next):

self.value = value        self.next =

next


class Linked_list:

   def __init__(self):
      self.head = None

   def insert_at_beginning(self,value):

self.head = Node(value,self.head)
```

```
    def insert_at_end(self,value):          if
self.head == None:
            self.head = Node(value,None)

        i = self.head









        while(i.next):
        i = i.next
              i.next = Node(value,None)


           def insert_at_specified_position(self,value,pos):
        if pos == 1:
                   node = Node(value,self.head)
        self.head = node                return


              i = self.head
        while(pos != 2):
        pos = pos - 1
        i = i.next
```

```
        node = Node(value,None)
node.next = i.next          i.next =
node


    def delete_from_beginning(self):
self.head = self.head.next


    def delete_from_end(self):
i = self.head          while(i.next
!= None):
            prev = i
i = i.next
prev.next = None
        del(i)

    def delete_at_specified_position(self,pos):
        ptr_del =
self.head
prev = self.head
i = self.head.next


        while(pos != 1):              pos -=
1               prev = ptr_del
ptr_del = i               i = i.next
prev.next = i             del(ptr_del)
def
delete_particular_element(self,val):
        i = self.head
ptr_del = self.head
```

```
prev = self.head
while(ptr_del.value !=
val):              prev =
ptr_del            ptr_del = i
i = i.next         prev.next
= i        del(ptr_del)
def
search_element(self,val):
i = self.head          count =
1        while(i.next !=
None):             if(i.value
== val):
               print("Element is Present at
position",count)                return
 else:
               count +=
1                i =
i.next
  print("Element not present in
linked list")         def
replace_element(self,pos,val):         i
= self.head
if pos == 1:
           i.value = val
return
       while(pos != 1):
pos = pos – 1
            i = i.next
i.value = val          def
reverse_traversal(self):
```

```
        i =
self.head
prev =
self.head
lst = []
while(i !=
None):
lst.append(i.value)
            i = i.next


        lst.reverse()
print("Reverse Traversal is : ")
for i in lst:
print(i,"-->",end = "")
print()

    #same as forward
traversal     def
display_list(self):
if self.head == None:
            print("Linked list is
Empty!!!")             return         i
= self.head          linked_list = ''
while(i):           linked_list +=
str(i.value) + '-->'             i =
i.next          print(linked_list)

    def insert_after_element(self,val,ele):
```

```
        i = self.head
    while(i.next != None):
    if(i.value == val):
            i.value = ele
print("value changed")
return     i = i.next
print("No such element
found")

if __name__ == '__main__':

   ll = Linked_list()
  print("Inserting elements in the beginning")
ll.insert_at_beginning(10)
ll.insert_at_beginning(200)
ll.insert_at_beginning(18)
ll.insert_at_beginning(30)      ll.display_list()
print()

   print("Inserting elements at the end")
ll.insert_at_end(20)     ll.insert_at_end(40)
ll.insert_at_end(50)     ll.insert_at_end(60)
ll.display_list()     print()

   print("Inserting element 100 at position 5")
ll.insert_at_specified_position(100,5)
ll.display_list()     print()
```

```
    print("Deleting the first element")
ll.delete_from_beginning()
ll.display_list()     print()

    print("Deleting last element")
ll.delete_from_end()
ll.display_list()     print()

    print("Deleting element from position 3")
ll.delete_at_specified_position(3)
ll.display_list()     print()

    print("Deleting element 20")
ll.delete_particular_element(20)
ll.display_list()     print()

    print("Searching element 18")     ll.search_element(18)      print()

    print("Replace  element  at  position
2")              ll.replace_element(2,238)
ll.display_list()     print()

    print("Reverse traversal of linked list")
ll.reverse_traversal()     print()

    print("Inserting element 258 in place of 238")    ll.insert_after_element(238,258)
ll.display_list()
```

Output:

```
C:\Python_Tsec>python Linkedlist.py
Inserting elements in the beginning
30-->18-->200-->10-->

Inserting elements at the end
30-->18-->200-->10-->20-->40-->50-->60-->

Inserting element 100 at position 5
30-->18-->200-->10-->100-->20-->40-->50-->60-->

Deleting the first element
18-->200-->10-->100-->20-->40-->50-->60-->

Deleting last element
18-->200-->10-->100-->20-->40-->50-->

Deleting element from position 3
18-->200-->100-->20-->40-->50-->

Deleting element 20
18-->200-->100-->40-->50-->

Searching element 18
Element is Present at position 1

Replace element at position 2
18-->238-->100-->40-->50-->

Reverse traversal of linked list
Reverse Traversal is :
50 -->40 -->100 -->238 -->18 -->

Inserting element 258 in place of 238
value changed
18-->258-->100-->40-->50-->
```

2. Python program to implement Stack data structure create class Stack with following functions.

- Push an element
- Pop an element
- Top of Stack
- Search an element
- Display stack create instance and perform all operations

Code:

```
#stack.py class Stack:
    def __init__(self):          self.st =
[]     def isempty(self):
return self.st == []
    def push(self,element):
self.st.append(element)
    def pop(self):
if self.isempty():
return -1
else:
            return self.st.pop()
def peek(self):
        n = len(self.st)
return self.st[n-1]      def
search(self,element):          if
self.isempty():
```

```
        return -1        else:
         try:
            n = self.st.index(element)
return len(self.st)-n            except
ValueError:
            return         -2     def
display(self):        return self.st
# Main Code: from stack import Stack s =
Stack() choice = 0 while choice<6:
print('STACK OPERATIONS')
print('1.Push element')      print('2.Pop
element')      print('3.Top of stack')
print('4.Search an element')
print('5.Display Stack')      print('6.Exit')
choice = int(input('Your choice: '))      if
choice==1:         element = int(input('Enter
element: '))        s.push(element)      elif
choice==2:         element = s.pop()         if
element == -1:
        print('The stack is empty')        else:
        print('Popped element= ', element)     elif
choice==3:
       element = s.peek()
print('Topmost element= ', element)      elif
choice==4:
      element = int(input('Enter element: '))
pos = s.search(element)         if pos == -1:
```

```
        print('The stack is empty')        elif
pos == -2:
        print('Element not found in the stack')        else:
        print('Element found at position: ', pos)     elif
choice==5:
      print('Stack',s.display())
else:         break
```

Output:

```
C:\Python_Tsec>python stack.py

C:\Python_Tsec>python assignment1.2.py
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 1
Enter element: 1
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 1
Enter element: 2
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 1
Enter element: 3
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 1
Enter element: 4
STACK OPERATIONS
```

```
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 5
Stack [1, 2, 3, 4]
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 2
Popped element=  4
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 3
Topmost element=  3
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 4
Enter element: 2
Element found at position:  2
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
```

```
4.Search an element
5.Display Stack
6.Exit
Your choice: 5
Stack [1, 2, 3]
STACK OPERATIONS
1.Push element
2.Pop element
3.Top of stack
4.Search an element
5.Display Stack
6.Exit
Your choice: 6
```

3. Python program to implement Queue data structure create class Queue with following functions.

- Insert an element
- Remove an element
- Search an element
- Display queue

Code:

```
#que1.py class Queue:
        def __init__(self):
         self.qu = []
    def isempty(self):
    return self.qu == []
       def add(self,element):
          self.qu.append(element)
    def delete(self):          if
    self.isempty():              return -1
    else:
              return self.qu.pop(0)
    def search(self,element):
    if self.isempty():
    return -1          else:
    try:
                  n = self.qu.index(element)                return n+1
             except ValueError:
    return   -2
     def              display(self):
```

return self.qu # Main
Code: from que1 import

Queue q = Queue() choice = 0

while choice<5:

print('QUEUE OPERATIONS')

print('1.Insert an element')

print('2.Remove an element')

print('3.Search an element')

print('4.Display Queue') print('5.Exit')

choice = int(input('Your choice: ')) if

choice==1: element =

int(input('Enter element: '))

q.add(element) elif choice==2:

element = q.delete() if

element == -1:

print('The queue is empty') else:

print('Removed element= ', element) elif choice==3:

element = int(input('Enter element: '))

pos = q.search(element) if pos == -1:

print('The queue is empty') elif pos == -2:

print('Element not found in the queue') else:

print('Element found at position: ', pos) elif
choice==4:

print('Queue',q.display())

else:

break

Output:

```
C:\Python_Tsec>python que1.py

C:\Python_Tsec>python assignment1.3.py
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 1
Enter element: 10
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 1
Enter element: 20
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 1
Enter element: 30
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 1
Enter element: 40
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
```

```
5.Exit
Your choice: 4
Queue [10, 20, 30, 40]
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 2
Removed element=  10
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 3
Enter element: 40
Element found at position:  3
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 4
Queue [20, 30, 40]
QUEUE OPERATIONS
1.Insert an element
2.Remove an element
3.Search an element
4.Display Queue
5.Exit
Your choice: 5
```

4. Python program to use deque class from collections with following functions.

- Add element at Front
- Remove element from Front
- Add element at Rear ● Remove element from Rear
- Search for an element

Code:

```
from collections import deque d = deque() choice =
0 while choice<6:
    print('DEQUE OPERATIONS')
    print('1.Insert an element at front')     print('2.Remove
an element from front')     print('3.Insert an element at rear')
print('4.Remove an element from rear')     print('5.Search an
element')


    print('6.Exit')     choice = int(input('Your
choice: '))     if choice==1:
        element = int(input('Enter element: '))
d.appendleft(element)     elif choice==2:
                if len(d) == 0:
            print('Deque is empty')         else:
            d.popleft()     elif choice==3:
        element = int(input('Enter element: '))
d.append(element)     elif choice==4:
```

```
            if len(d) == 0:
        print('Deque is empty')        else:
        d.pop()     elif choice==5:
      element = int(input('Enter element: '))        c =
d.count(element)        print('No of times the element
found:', c)     else:        break     print('Deque=' , end='')
for i in d:        print(i, '
', end='')     print()
```

Output:

```
C:\Python_Tsec>python assignment1.4.py
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 1
Enter element: 11
Deque=11
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 1
Enter element: 22
Deque=22  11
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 3
Enter element: 33
Deque=22  11  33
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 3
Enter element: 44
```

```
Your choice: 3
Enter element: 44
Deque=22  11  33  44
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 2
Deque=11  33  44
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 4
Deque=11  33
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 5
Enter element: 33
No of times the element found: 1
Deque=11  33
DEQUE OPERATIONS
1.Insert an element at front
2.Remove an element from front
3.Insert an element at rear
4.Remove an element from rear
5.Search an element
6.Exit
Your choice: 6
```