Code:

```
print("**************************************************")
print("MENU DRIVEN CODE FOR DATA STRUCTURES")
print("**************************************************")
print("__________________________________________________")
while(1):
      print("Choose from the menu")
      print("1. Linked List Operations")
      print("2. Stack Operations")
      print("3. Queue Operations")
      print("4. Exit")
      ch=int(input("Enter your Choice: "))
      print("__________________________________________________")

      if ch==1:
      class Node:
            def __init__(self, data):
                  self.item = data
                  self.ref = None
      class LinkedList:
            def __init__(self):
                  self.start_node = None
            def create(self):
                  nums = int(input("How many nodes do you want to create: "))
                  if nums == 0:
                  return
                  for i in range(nums):
                  value = int(input("Enter the value for the node: "))
                  self.insert_at_end(value)
            def traverse(self):
                  if self.start_node is None:
                  print("List has no element!!!")
                  return
                  else:
                  n = self.start_node
                  while n is not None:
                        print(n.item , " ")
                        n = n.ref
            def insert(self, index, data):
```

```
        if index == 1:
        new_node = Node(data)
        new_node.ref = self.start_node
        self.start_node = new_node
        i = 1
        n = self.start_node
        while i < index-1 and n is not None:
        n = n.ref
        i = i+1
        if n is None:
        print("Index out of bound")
        else:
        new_node = Node(data)
        new_node.ref = n.ref
        n.ref = new_node

    def search(self, x):
        if self.start_node is None:
        print("List has no elements")
        return
        n = self.start_node
        while n is not None:
        if n.item == x:
            print("Item found")
            return True
        n = n.ref
        print("item not found")
        return False
    def delete(self, x):
        if self.start_node is None:
        print("The list has no element to delete")
        return

        if self.start_node.item == x:
        self.start_node = self.start_node.ref
        return

        n = self.start_node
        while n.ref is not None:
        if n.ref.item == x:
            break
        n = n.ref

        if n.ref is None:
```

```
            print("item not found in the list")
            else:
            n.ref = n.ref.ref
      def insert_at_end(self, data):
            new_node = Node(data)
            if self.start_node is None:
            self.start_node = new_node
            return
            n = self.start_node
            while n.ref is not None:
            n= n.ref
            n.ref = new_node;
ll = LinkedList()



choice=0
while choice<5:
      print("_________________________")
      print("LINKED LIST OPERATIONS")
      print("_________________________")
      print("1. Insert elements")
      print("2. Remove element")
      print("3. Search for element")
      print("4. Create a linked list")
      print("5. Exit")
      choice=int(input("Enter your choice: "))
      print("_______________________________")

      if choice==1:
            index= int(input("Enter the index: "))
            ele=input("Enter the element")
            ll.insert(index, ele)
            ll.traverse()

      elif choice==2:
            element=int(input("Enter element to be deleted: "))
            ll.delete(element)
            ll.traverse()
      elif choice==3:
            element = int(input("Enter element to be searched: "))
            result=ll.search(element)
            print(result)
      elif choice==4:
```

```
            ll.create()
            ll.traverse()
      else:
            break



elif(ch==2):
class Stack:
      def __init__(self):
            self.st = []

      def isEmpty(self):
            return self.st == []

      def push(self, item):
            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

s=Stack()
choice=0
while(choice<5):
      print("_________________________")
```

```
        print("STACK OPERATIONS")
        print("_________________________")
        print("1. Push element")
        print("2. Pop element")
        print("3. Peek element")
        print("4. Search for element")
        print("5. Exit")
        choice=int(input("Enter your choice: "))
        print("_______________________________")

        if choice==1:
              element = int(input('Enter element to be inserted: '))
              s.push(element)
        elif choice==2:
              element=s.pop()
              if element == -1:
              print("The stack is empty!!!")
              else:
              print("Popped elements=", element)
        elif choice==3:
              element = s.peek()
              print("Topmost element=",element)
        elif choice==4:
              element = int(input("Enter element to be found: "))
              pos = s.search(element)
              if pos == -1:
              print("Stack is empty")
              elif pos==-2:
              print("Element not found")
              else:
              print("Element found at postion:",pos)
        else:
              break
        print('Stack=',s.display())
elif(ch==3):
class Queue:
        def __init__(self):
              self.qu=[]
        def isempty(self):
              return self.qu==[]
        def enqueue(self, element):
              self.qu.append(element)
        def dequeue(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
q=Queue()
choice=0
while choice<4:
      print("_________________________")
      print("QUEUE OPERATIONS")
      print("_________________________")
      print("1. Enqueue element")
      print("2. Delete element")
      print("3. Search for element")
      print("4. Exit")
      choice=int(input("Enter your choice: "))
      print("_______________________________")

      if choice==1:
            element=float(input("Enter element to be inserted: "))
            q.enqueue(element)
      elif choice==2:
            element=q.dequeue()
            if element == -1:
            print("The queue is empty!!!")
            else:
            print("Removed element=",element)
      elif choice==3:
            element=float(input("Enter element to be found: "))
            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)
      else:
            break
      print("Queue= ",q.display())
elif(ch==4):
break
else:
print("Invalid choice, please choose again!!!")
```

Output:

```
************************************************
     MENU DRIVEN CODE FOR DATA STRUCTURES
************************************************

_____________________________________________________
Choose from the menu
1. Linked List Operations
2. Stack Operations
3. Queue Operations
4. Exit
Enter your Choice: 1
_____________________________________________________

_____________________________
LINKED LIST OPERATIONS

_____________________________
1. Insert elements
2. Remove element
3. Search for element
4. Create a linked list
5. Exit
Enter your choice: 4

_____________________________________
How many nodes do you want to create: 3
Enter the value for the node: 25
Enter the value for the node: 50
Enter the value for the node: 75
25
50
75

_____________________________
LINKED LIST OPERATIONS

_____________________________
1. Insert elements
2. Remove element
3. Search for element
4. Create a linked list
5. Exit
Enter your choice: 1

_____________________________________
Enter the index: 3
Enter the element55
25
50
55
75

_____________________________
LINKED LIST OPERATIONS

_____________________________
1. Insert elements
2. Remove element
3. Search for element
4. Create a linked list
5. Exit
Enter your choice: 2

_____________________________________
Enter element to be deleted: 50
25
55
```

75

```
_____________________________
LINKED LIST OPERATIONS

_____________________________
1. Insert elements
2. Remove element
3. Search for element
4. Create a linked list
5. Exit
Enter your choice: 3

_____________________________________
Enter element to be searched: 75
Item found
True

_____________________________
LINKED LIST OPERATIONS

_____________________________
1. Insert elements
2. Remove element
3. Search for element
4. Create a linked list
5. Exit
Enter your choice: 5

_____________________________________
Choose from the menu
1. Linked List Operations
2. Stack Operations
3. Queue Operations
4. Exit
Enter your Choice: 2

____________________________________________________________

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 23
Stack= [23]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 34

_____________________________________
Choose from the menu
1. Linked List Operations
2. Stack Operations
3. Queue Operations
4. Exit
Enter your Choice: 45

____________________________________________________________
```

```
Invalid choice, please choose again!!!
Choose from the menu
1. Linked List Operations
2. Stack Operations
3. Queue Operations
4. Exit
Enter your Choice: 2
_____________________________________________________
_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 23
Stack= [23]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 34
Stack= [23, 34]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 45
Stack= [23, 34, 45]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 2

_____________________________________
Popped elements= 45
Stack= [23, 34]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
```

```
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 3

_____________________________________
Topmost element= 34
Stack= [23, 34]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 55
Stack= [23, 34, 55]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 4

_____________________________________
Enter element to be found: 34
Element found at postion: 2
Stack= [23, 34, 55]

_____________________________
STACK OPERATIONS

_____________________________
1. Push element
2. Pop element
3. Peek element
4. Search for element
5. Exit
Enter your choice: 5

_____________________________________
Choose from the menu
1. Linked List Operations
2. Stack Operations
3. Queue Operations
4. Exit
Enter your Choice: 3

_____________________________________________________________

_____________________________
QUEUE OPERATIONS

_____________________________
1. Enqueue element
2. Delete element
3. Search for element
4. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 23
```

```
Queue=  [23.0]

_____________________________
QUEUE OPERATIONS

_____________________________
1. Enqueue element
2. Delete element
3. Search for element
4. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 34
Queue=  [23.0, 34.0]

_____________________________
QUEUE OPERATIONS

_____________________________
1. Enqueue element
2. Delete element
3. Search for element
4. Exit
Enter your choice: 1

_____________________________________
Enter element to be inserted: 45
Queue=  [23.0, 34.0, 45.0]

_____________________________
QUEUE OPERATIONS

_____________________________
1. Enqueue element
2. Delete element
3. Search for element
4. Exit
Enter your choice: 3

_____________________________________
Enter element to be found: 34
Element found at position:  2
Queue=  [23.0, 34.0, 45.0]

_____________________________
QUEUE OPERATIONS

_____________________________
1. Enqueue element
2. Delete element
3. Search for element
4. Exit
Enter your choice: 2

_____________________________________
Removed element= 23.0
Queue=  [34.0, 45.0]

_____________________________
QUEUE OPERATIONS

_____________________________
1. Enqueue element
2. Delete element
3. Search for element
4. Exit
Enter your choice: 4

_____________________________________
Choose from the menu
1. Linked List Operations
2. Stack Operations
3. Queue Operations
4. Exit
```

```
Enter your Choice: 4
```