16) Write a CPP program to perform operations on singly linked list.

Ans: Source code

```
#include <iostream.h>
#include <process.h>
#include <conio.h>
class node
{
  public :
  int data;
  node * next;
};
class list
{
  private :
  node *head;
  public :
  list().
  {
  head = Null;
}
void insertbeg(int item);
void display();
void insertend(int item);
```

```
void intermediate (int key, int item);
void deletebeg ();
void deleteend ();
void deleteintermediate (int key);
};
Void list:: insertbeg (int item)
{
    node * newnode = newnode();
    if (head == NULL)
    {
        head = newnode;
        newnode -> data = item;
        newnode -> next = NULL;
    }
    else
    {
        newnode ->data = item;
        newnode -> next = head;
        head = newnode;
    }
}
Void list:: insertend (int item)
{
    node * temp = new node();
    node * newnode = new node();
```

```
if (head == NULL)
{
    newnode->data = item;
    newnode->next = NULL;
    head = newnode;
}
else
{
    temp = head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    newnode->next = NULL;
    newnode->data = item;
    temp->next = newnode;
}
}

Void list :: intermediate (int key, int item)
{
    node * temp = new node();
    temp = head;
    if (head == NULL)
    {
        node * newnode = new node();
        newnode->data = item;
        newnode->next = NULL;
```

```
        head = new node ;
    }
    else
    {
      while (temp1 = NULL && temp->data != key)
      {
          temp = temp ->next ;
      }
    if(temp = = NULL)
    {
     cout<< "\n key not found. Insertion operation, cannot
    performed .\n";
     return;
    }
    node * newnode = new node();
    newnode ->next = temp->next ;
    newnode -> data = item;
    temp->next = newnode ;
    }
}
void list :: delete beg ()
{
    node * temp = new node();
    if (head = = NULL)
    {
      cout << "\n list is empty \n";
      return;
```

```
    }
    else
    {
      temp = head ;
      if ( temp -> next == NULL)
      {
         head = NULL;
      }
      else
      {
        head = temp->next ;
      }
      cout << "\n deleted node is " << temp -> data ;
      delete(temp) ;
    }
}
void list :: deleteend()
{
    if(head == NULL)
    {
      cout<< "\n list is empty \n";
      return;
    }
node * temp1 = head;
node * temp2 = NULL;
```

```
while (temp1 -> next != NULL)
{
    temp2 = temp1;
    temp1 = temp1 -> next;
}
if( temp2 == NULL)
{
    head = NULL;
}
else
{
    temp2->next = NULL;
}
cout<< "\n deleted node is "<< temp1->data;
delete(temp1);
}
Void list :: deleteintermediate (int key)
{
    if(head == NULL)
    {
    cout<< "\n list is empty \n";
    return;
    }
node * temp1 = head;
node * temp2 = NULL;
```

```
while ( temp1 != NULL && temp1 -> data != key)
{
   temp2 = temp1;
   temp1 = temp1 -> next;
}
if ( temp1 == NULL)
{
   cout << "\n key not found. Deletion operation cannot be
performed .\n";
   return;
}
if ( temp2 == NULL)
{
   head = temp1 -> next ;
}
else
{
 temp2 -> next = temp1 -> next;
}
cout << "\n deleted node is " << temp1 -> data
delete (temp1);
}
void list :: display()
{
 if ( head == NULL)
 {
```

```
        cout<<"\n list is empty. cannot display.";
    }
    else
    {
        node *temp = head;
        while(temp != NULL)
        {
            cout<< temp->data <<"\t";
            temp = temp->next;
        }
    }
}
void main()
{
    list obj;
    int num, choic, key;
    do
{
```

cout<<"\n 1. insertion at the beginning \n 2. Insertion at end of the list \n 3. Insertion at intermediate \n 4. Deletion at beginning \n 5. Deletion at end \n 6. Deletion at intermediate \n 7. display \n 8. Exit \n";

```
cout<<"\n enter your choice.:";
cin>>choice;
Switch(choice)
```

```
{
case 1 :
   cout<<"\n enter the number to be inserted :";
   cin>>num;
   obj.insertbeg(num);
   break;
case 2 :
   cout<<"\n enter the number to be inserted :";
   cin>>num;
   obj.insertend(num);
   break;
case 3 :
    cout<<"\n enter the key to be inserted :";
    cin>> key;
    cout<<"\n enter the element to be inserted :";
   cin>>num;
    obj.intermediate(key, num);
    break;
case 4 :
    obj.deletebeg();
    break;
case 5 :
   obj.deleteend();
   break;
case 6 :
    cout<<"\n enter the node to be deleted :";
    cin>>num;
```

```
            obj.deleteintermediate(num);
            break;
        case 7:
            obj.display();
            break;
        case 8:
            exit(0);
            break;
        default:
          cout<<"\n invalid choice :";
        }
    }while(choice != 8);
    getch();
}
```

Output

1. Insertion at beginning.
2. Insertion at end of the list.
3. Insertion at intermediate
4. Deletion at beginning
5. Deletion at end
6. Deletion at intermediate
7. Display
8. exit

Date : .....

Page No:

enter your choice : 1

enter the number to be inserted : 12.

1. Insertion at beginning
2. Insertion at end of a list
3. Insertion at intermediate
4. Deletion at beginning
5. Deletion at end
6. Deletion at intermediate
7. Display
8. exit.

enter your choice : 3

Enter the key to be inserted : 12

enter the element to be inserted : 55

1. Insertion at beginning
2. Insertion at end of a list
3. Insertion at intermediate
4. Deletion at beginning
5. Deletion at end
6. Deletion at intermediate
7. Display
8. exit.

enter your choice : 6

Deleted node is 55

1. Insertion at beginning
2. Insertion at end of a list
3. Insertion at intermediate.

4. Deletion at begining
5. Deletion at end
6. Deletion at intermediate
7. Display
8. exit.
enter your choice: 6
enter the node to be deleted : 12
deleted node is 12.
1. insertion at begining
2. insertion at end of the list
3. insertion at intermediate
4. Deletion at begining
5. Deletion at end
6. Deletion at intermediate
7. Display
8. exit
enter your choice: 8.