Assignment No.03

```
#include<iostream>
using namespace std;
class node
{
   int data;
   node*next;
   node*pre;
   public:
   node*create(node*);
   void insbeg(node*);
   void insend(node*);
   void inspos(node*);
   void delbeg(node*);
   void delend(node*);
   void delpos(node*);
   void display(node*);
   void reverse(node*);
   void search(node*);
};
node*node::create(node*head)
{
   head=new node;
   head->next=NULL;
   head->pre=NULL;
   return head;
}
void node::insbeg(node*head)
{
            node* temp;
```

```
        temp=new node;
        cout<<"Enter data (beginning) : ";
        cin>>temp->data;
        temp->next=NULL;
        temp->pre=NULL;
        if(head->next==NULL)
        {
        temp->pre=head;
        head->next=temp;
        }
        else
        {
                temp->next=head->next;
                temp->pre=head;
                head->next=temp;
        }
}
void node::display(node*head)
{
        node *cn=head->next;
        while(cn!=NULL)
        {
                cout<<cn->data<<"--->";
                cn=cn->next;
        }
}
void node::insend(node*head)
{
        node*nn = new node;
        cout << "\n Enter data(end) : ";
        cin >> nn->data;
```

```
        node *cn;
        cn = head;
        nn->next=NULL;
        nn->pre=NULL;
        if(head==NULL)
        {
           head=nn;
            nn->next=head;
         }
          while (cn->next != NULL)
         {
                cn = cn->next;
        }
        cn->next = nn;
        nn->pre=cn;
}
void node::inspos(node*head)
{
        int pos=0,count=0;
        cout<<"\nEnter position :";
        cin>>pos;
        node*cn=head;
        while(cn!=NULL)
        {
                count++;
                cn=cn->next;
        }
         if(pos==1)
        {
                 insbeg(head);
        }
```

```
        else if(pos==count)
        {
                insend(head);
        }
        else
        {
                node* temp=head;
                node *cn=NULL;
                node*nn=new node;
                cout<<"\n Enter data (position) : ";
                cin>>nn->data;
                for(int i=0;i< pos;i++)
                {
                cn=temp;
                temp=temp->next;
                 }
                nn->next=temp;
                temp->pre=nn;
                cn->next=nn;
                nn->pre=cn;
         }
}
void node::delbeg(node*head)
{
        node*curr;
        curr=head->next;
        if(head->next==NULL)
         {
                        cout<<"\nCan't Delete...";
        }
         else
```

```
        {
                head->next=curr->next;
                head->pre=NULL;
                delete curr;
                cout<<"\nDeleted sucessfully.....\n";
        }
}
void node::delend(node*head)
{
        node *temp,*cn=head;
        while(cn->next!=NULL)
        {
                temp=cn;
                cn=cn->next;
        }
        cn->pre->next=cn->next;
        cn->next=NULL;
        delete cn;
        cout<<"\nDeleted sucessfully.....\n";
}
void node::delpos(node*head)
{
        int pos,count=1;
        cout<<"\n Enter position to delete : ";
         cin>>pos;
        node *cn=head;
        while(cn!=NULL)
        {
                count++;
   cn=cn->next;
  }
```

```
        if(pos==1)
        {
                delbeg(head);
        }
        else if(pos==count)
        {
                delend(head);
        }
        else
        {
                cn=head;
                node *temp;
                for(int i=0;i<pos;i++)
                {
                        temp=cn;
                        cn=cn->next;
                }
                temp->next=cn->next;
                cn->pre=temp;
                delete cn;
                cout<<"\nDeleted sucessfully.....\n";
        }
}
void node::reverse(node*head)
{
        node *cn=head->next;
        while(cn->next!=NULL)
        {
                cn=cn->next;
        }
        while(cn!= head)
```

```
        {
                cout<<cn->data<<"--->";
                cn=cn->pre;
        }
}
void node::search(node*head)
{
        int s,count=0,f=0;
        node *cn=head;
        cout<<"\n Enter data to search:";
        cin>>s;
        while(cn!=NULL)
        {
                if(cn->data==s)
                {
                        cout<<"\n Element found at "<<count;
                        f=0;
                        break;
                }
                 else
                {
                        f=1;
                }
                count++;
                cn=cn->next;
        }
        if(f==1)
        {
                cout<<"\n Element not found";
        }
}
```

```
int main()
{
    node obj;
    node*head;
    head=obj.create(head);
    int ch;
    while(1)
    {
        cout<<"\n----------------------------------------------------------------";
        cout<<"\nYOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST ";
        cout<<"\n----------------------------------------------------------------";
        cout<<"\n1.Insert at beginning";
        cout<<"\n2.Insert at end";
        cout<<"\n3.Insert at middle";
        cout<<"\n4.Display forward";
        cout<<"\n5.Display backward";
        cout<<"\n6.Search";
        cout<<"\n7.Delete at beginning";
        cout<<"\n8.Delete at end";
        cout<<"\n9.Delete at middle";
        cout<<"\n10.Exit ";
        cout<<"\n----------------------------------------------------------------";
        cout<<"\nEnter choice:";
        cin>>ch;
        switch(ch)
        {
            case 1:obj.insbeg(head);
                   break;
            case 2:obj.insend(head);
                   break;
            case 3:obj.inspos(head);
```

```
            break;
        case 4:cout<<"\nDisplay forward.....\n";
            obj.display(head);
            break;
        case 5:cout<<"\nDisplay backward...\n";
            obj.reverse(head);
            break;
        case 6:obj.search(head);
            break;
        case 7:obj.delbeg(head);
            break;
        case 8:obj.delend(head);
            break;
        case 9:obj.delpos(head);
            break;
        case 10:exit(0);
      }
   }return 0;
}
```

Output:

```
------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
------------------------------------------------------------
Enter choice:1
Enter data (beginning) : 45
```

```
------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
------------------------------------------------------------
Enter choice:2

 Enter data(end) : 33
```

```
------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
------------------------------------------------------------
Enter choice:2

 Enter data(end) : 67
```

```
------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
------------------------------------------------------------
Enter choice:2

 Enter data(end) : 88
```

```
------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
------------------------------------------------------------
Enter choice:3

Enter position :3

 Enter data (position) : 15
```

```
------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
------------------------------------------------------------
Enter choice:6

 Enter data to search:15

 Element found at 3
------------------------------------------------------------
```

```
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:4

Display forward.....
45--->33--->15--->67--->88--->
----------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
----------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:5

Display backward...
88--->67--->15--->33--->45--->
```

```
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:7

Deleted sucessfully........

----------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
----------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:4

Display forward.....
33--->15--->67--->88--->
```

```
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:8

Deleted sucessfully..........

----------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
----------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:4

Display forward.....
33--->15--->67--->
```

```
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:9

 Enter position to delete : 3

Deleted sucessfully.......

----------------------------------------------------------------
YOU CAN PERFORM FOLLOWING OPREATIONS USING DOUBLY LINKED LIST
----------------------------------------------------------------
1.Insert at beginning
2.Insert at end
3.Insert at middle
4.Display forward
5.Display backward
6.Search
7.Delete at beginning
8.Delete at end
9.Delete at middle
10.Exit
----------------------------------------------------------------
Enter choice:4

Display forward.....
33--->15--->
```