```cpp
// Singly Linked List

#include<iostream>
using namespace std;

class Node
{
    public:
    int x;
    Node* next; //pointer of node

    Node(int val) //constructor
    {
        x = val;
        next = NULL;
    }
};

int main()
{
//static object
    Node box1(1);
    Node box2(2);
    Node box3(3);

    box1.next = &box2; // address of box2
    box2.next = &box3;
    box3.next = NULL;

    cout << box1.x << endl; //box1
    cout << (*box1.next).x << endl; //box2
    cout << (*(*box1.next).next).x << endl; //box3


//dynamic object
    Node* box4 = new Node(4);
    Node* box5 = new Node(5);
    Node* box6 = new Node(6);

    box4->next = box5;
    box5->next = box6;
    box6->next = NULL;

    cout << box4->x << endl; //box4
    cout << box4->next->x << endl; //box5
    cout << box4->next->next->x << endl; //box6

}

```

```cpp
// Singly Linked List (Insertion and Deletion)

#include<iostream>
using namespace std;

class Node
{
public:
    int val; // value
    Node* next; // next address

    Node(int x) //constructor
    {
        val = x;
        next = NULL;
    }
};

void insert_tail(Node* &head, Node* &tail, int val)
{
    if(head==NULL)
    {
        head = tail = new Node(val);
        return;
    }
    tail = tail->next = new Node(val);
}

void insert_head(Node* &head, Node* &tail, int val)
{
    if(head==NULL)
    {
        head = tail = new Node(val);
        return;
    }
    Node* tmp = head; //old head
    head = new Node(val); //new head
    head->next = tmp;
}

void insert_at(Node* &head, int index, int val)
{
    if(index == 0)
    {
        Node* tmp = head; //old head
        head = new Node(val); //new head
        head->next = tmp;
        return;
    }
    Node* tmp = head;
    for(int i=1; i<index; i++)
    {
        tmp = tmp->next;
        if(tmp == NULL)
        {
            cout<<"Invalid index!\n";
```

```
57                  return;
58              }
59          }
60          Node* a = new Node(val);
61          a->next = tmp->next;
62          tmp->next = a;
63      }
64
65      void delete_head(Node* &head, Node* &tail)
66      {
67          if(head == NULL)
68          {
69              cout<<"empty!\n";
70              return;
71          }
72          Node* tmp = head;
73          head = head->next;
74          delete tmp;
75      }
76
77      void delete_at(Node* &head, Node* &tail, int index)
78      {
79          if(head == NULL)
80          {
81              cout<<"empty!\n";
82              return;
83          }
84          Node* tmp = head;
85          for(int i=1; i<index; i++)
86          {
87              tmp = tmp->next;
88              if(tmp == NULL || tmp->next == NULL)
89              {
90                  cout<<"Invalid index!\n";
91                  return;
92              }
93          }
94          Node* dlt = tmp->next;
95          tmp->next = tmp->next->next;
96          delete dlt;
97      }
98
99      void print(Node* head)
100     {
101         Node* tmp = head;
102         while(tmp != NULL)
103         {
104             cout<<tmp->val<<" ";
105             tmp = tmp->next;
106         }
107         cout<<endl;
108     }
109
110
111     int main()
112     {
```

```
113       cout<<"1. insert at tail\n";
114       cout<<"2. insert at head\n";
115       cout<<"3. insert at position\n";
116       cout<<"4. delete head\n";
117       cout<<"5. delete from position\n";
118       cout<<"6. print list\n";
119       cout<<"press any key to end\n";
120
121       Node* head = NULL; //initially
122       Node* tail = NULL; //initially
123
124       while(true)
125       {
126           int command; cin>>command;
127           if(command == 1)
128           {
129               int x; cin>>x;
130               insert_tail(head,tail,x);
131           }
132           else if(command == 2)
133           {
134               int x; cin>>x;
135               insert_head(head,tail,x);
136           }
137           else if(command == 3)
138           {
139               int i,x; cin>>i>>x;
140               insert_at(head,i,x);
141           }
142           else if(command == 4)
143           {
144               delete_head(head,tail);
145           }
146           else if(command == 5)
147           {
148               int i; cin>>i;
149               delete_at(head,tail,i);
150           }
151           else if(command == 6)
152           {
153               print(head);
154           }
155           else
156           {
157               cout<<"end\n";
158               break;
159           }
160       }
161
162   return 0;
163   }
164
```

```cpp
// Doubly Linked List

#include<iostream>
using namespace std;

class Node
{
    public:
    int x;
    Node* next; //pointer of node
    Node* prev; //pointer of node

    Node(int val) //constructor
    {
        x = val;
        next = NULL;
        prev = NULL;
    }
};

int main()
{
//static object
    Node box1(1);
    Node box2(2);
    Node box3(3);

//links
    box1.prev = NULL;
    box1.next = &box2;
    box2.prev = &box1;
    box2.next = &box3;
    box3.prev = &box2;
    box3.next = NULL;

    cout << box1.x << " "; //box1
    cout << (*box1.next).x << " "; //box2
    cout << (*(*box1.next).next).x << endl; //box3

//print int reverse order
    cout << box3.x << " "; //box3
    cout << (*box3.prev).x << " "; //box2
    cout << (*(*box3.prev).prev).x << endl; //box1


//dynamic object
    Node* box4 = new Node(4);
    Node* box5 = new Node(5);
    Node* box6 = new Node(6);

//links
    box4->prev = NULL;
    box4->next = box5;
    box5->prev = box4;
    box5->next = box6;
    box6->prev = box5;
```

```
57        box6->next = NULL;
58
59        cout << box4->x << " "; //box4
60        cout << box4->next->x << " "; //box5
61        cout << box4->next->next->x << endl; //box6
62
63    //print int reverse order
64        cout << box6->x << " "; //box6
65        cout << box6->prev->x << " "; //box5
66        cout << box6->prev->prev->x << endl; //box4
67
68    }
69
```

```cpp
// Doubly Linked List (Insertion)

#include<iostream>
using namespace std;

class Node
{
public:
    int val;
    Node *next;
    Node *prev;
    Node(int val)
    {
        this->val = val;
        this->next = NULL;
        this->prev = NULL;
    }
};

void print_normal(Node *head)
{
    Node *tmp = head;
    while (tmp != NULL)
    {
        cout << tmp->val << " ";
        tmp = tmp->next;
    }
    cout << endl;
}
void print_reverse(Node *tail)
{
    Node *tmp = tail;
    while (tmp != NULL)
    {
        cout << tmp->val << " ";
        tmp = tmp->prev;
    }
    cout << endl;
}

void insert_at_position(Node *head, int pos, int val)
{
    Node *newNode = new Node(val);
    Node *tmp = head;
    for (int i = 1; i <= pos - 1; i++)
    {
        tmp = tmp->next;
    }
    newNode->next = tmp->next;
    tmp->next = newNode;
    newNode->next->prev = newNode;
    newNode->prev = tmp;
}

int size(Node *head)
{
```

```
57        Node *tmp = head;
58        int cnt = 0;
59        while (tmp != NULL)
60        {
61            cnt++;
62            tmp = tmp->next;
63        }
64        return cnt;
65    }
66
67    void insert_head(Node *&head, Node *&tail, int val)
68    {
69        Node *newNode = new Node(val);
70        if (head == NULL)
71        {
72            head = newNode;
73            tail = newNode;
74            return;
75        }
76        newNode->next = head;
77        head->prev = newNode;
78        head = newNode;
79    }
80
81    void insert_tail(Node *&head, Node *&tail, int val)
82    {
83        Node *newNode = new Node(val);
84        if (tail == NULL)
85        {
86            head = newNode;
87            tail = newNode;
88            return;
89        }
90        tail->next = newNode;
91        newNode->prev = tail;
92        tail = tail->next;
93    }
94
95
96    int main()
97    {
98        Node *head = new Node(10);
99        Node *a = new Node(20);
100       Node *b = new Node(30);
101       Node *c = new Node(40);
102       Node *tail = c;
103
104   // connection
105       head->next = a;
106       a->prev = head;
107       a->next = b;
108       b->prev = a;
109       b->next = c;
110       c->prev = b;
111
112       int pos, val;
```

```
113         cin >> pos >> val;
114
115         if (pos > size(head))
116         {
117             cout << "Invalid" << endl;
118         }
119         else if (pos == 0)
120         {
121             insert_head(head, tail, val);
122         }
123         else if (pos == size(head))
124         {
125             insert_tail(head, tail, val);
126         }
127         else
128         {
129             insert_at_position(head, pos, val);
130         }
131         print_normal(head);
132         print_reverse(tail);
133
134     return 0;
135     }
136
```

```cpp
// Doubly Linked List (Deletion)

#include<iostream>
using namespace std;

class Node
{
public:
    int val;
    Node *next;
    Node *prev;
    Node(int val)
    {
        this->val = val;
        this->next = NULL;
        this->prev = NULL;
    }
};

void print_normal(Node *head)
{
    Node *tmp = head;
    while (tmp != NULL)
    {
        cout << tmp->val << " ";
        tmp = tmp->next;
    }
    cout << endl;
}

void print_reverse(Node *tail)
{
    Node *tmp = tail;
    while (tmp != NULL)
    {
        cout << tmp->val << " ";
        tmp = tmp->prev;
    }
    cout << endl;
}

int size(Node *head)
{
    Node *tmp = head;
    int cnt = 0;
    while (tmp != NULL)
    {
        cnt++;
        tmp = tmp->next;
    }
    return cnt;
}

void delete_at_position(Node *head, int pos)
{
    Node *tmp = head;
```

```
57        for (int i = 1; i <= pos - 1; i++)
58        {
59            tmp = tmp->next;
60        }
61        Node *deleteNode = tmp->next;
62        tmp->next = tmp->next->next;
63        tmp->next->prev = tmp;
64        delete deleteNode;
65    }
66
67    void delete_tail(Node *&head, Node *&tail)
68    {
69        Node *deleteNode = tail;
70        tail = tail->prev;
71        delete deleteNode;
72        if (tail == NULL)
73        {
74            head = NULL;
75            return;
76        }
77        tail->next = NULL;
78    }
79
80    void delete_head(Node *&head, Node *&tail)
81    {
82        Node *deleteNode = head;
83        head = head->next;
84        delete deleteNode;
85        if (head == NULL)
86        {
87            tail = NULL;
88            return;
89        }
90        head->prev = NULL;
91    }
92
93    int main()
94    {
95        Node *head = new Node(10);
96        Node *a = new Node(20);
97        Node *b = new Node(30);
98        Node *c = new Node(40);
99        Node *tail = c;
100
101   // connection
102       head->next = a;
103       a->prev = head;
104       a->next = b;
105       b->prev = a;
106       b->next = c;
107       c->prev = b;
108
109       int pos;
110       cin >> pos;
111
112       if (pos >= size(head))
```

```
113         {
114             cout << "Invalid" << endl;
115         }
116         else if (pos == 0)
117         {
118             delete_head(head, tail);
119         }
120         else if (pos == size(head) - 1)
121         {
122             delete_tail(head, tail);
123         }
124         else
125         {
126             delete_at_position(head, pos);
127         }
128 
129         print_normal(head);
130         print_reverse(tail);
131 
132     return 0;
133     }
134 
```