

```

1  #include <iostream>
2
3  using namespace std;
4
5  struct Node {
6      int data;
7      Node* next = nullptr;;
8  };
9
10 Node* head = nullptr;
11
12 Node* newNode(int data) {
13     Node* node = new Node;
14     node->data = data;
15     return node;
16 }
17
18
19 void insertAtHead(int data) {
20     Node* node = newNode(data);
21     if(head == nullptr) {
22         head = node;
23         return;
24     }
25     node->next = head;
26     head = node;
27 }
28
29
30 void insertAtTail(int data) {
31     if(head == nullptr) {
32         insertAtHead(data);
33         return;
34     }
35     Node* tail = head;
36     while(tail != nullptr) {
37         if (tail->next == nullptr) {
38             Node* node = newNode(data);
39             tail->next = node;
40             break;
41         }
42         tail = tail->next;
43     }
44 }
45
46
47 void insertAtMiddleAfter(int after, int data) {
48     Node* prev = head;
49     while(prev != nullptr) {
50         if(prev->data == after) {
51             Node* node = newNode(data);
52             node->next = prev->next;
53             prev->next = node;
54             break;
55         }
56         prev = prev->next;
57     }
58 }
59
60
61 void insertAtMiddleBefore(int before, int data) {
62     if(head != nullptr && head->data == before) {
63         insertAtHead(data);
64         return;
65     }
66
67     Node* prev = head;
68     while(prev != nullptr && prev->next != nullptr) {
69         if(prev->next->data == before) {
70             Node* node = newNode(data);
71             node->next = prev->next;
72             prev->next = node;
73             break;
74         }
75         prev = prev->next;
76     }
77 }
78
79 void deleteFromHead() {
80     if(head == nullptr) {
81         return;
82     }
83
84     Node* node = head;
85     head = node->next;
86     delete node;
87 }
88
89
90 void deleteFromTail() {
91     if(head == nullptr) {
92         return;
93     }
94     Node* prev = head;
95     while(prev != nullptr && prev->next != nullptr) {
96         if(prev->next->next == nullptr) {
97             Node* node = prev->next;
98             prev->next = nullptr;
99             delete node;
100            break;
101        }
102        prev = prev->next;
103    }
104 }
105
106

```

```

107 void deleteFromMiddle(int data) {
108     if (head != nullptr && head->data == data) {
109         deleteFromHead();
110         return;
111     }
112     Node* prev = head;
113     while (prev != nullptr && prev->next != nullptr) {
114         if (prev->next->data == data) {
115             Node* node = prev->next;
116             prev->next = prev->next->next;
117             delete node;
118             break;
119         }
120         prev = prev->next;
121     }
122 }
123
124 void showForward() {
125     Node* node = head;
126     while (node != nullptr) {
127         cout << " " << node->data;
128         node = node->next;
129     }
130     cout << endl;
131 }
132
133 int main() {
134     insertAtHead(10);
135     insertAtHead(20);
136     insertAtHead(30);
137     insertAtHead(40);
138     insertAtHead(50);
139     insertAtHead(60);
140     insertAtHead(70);
141     insertAtHead(80);
142     insertAtHead(90);
143     insertAtHead(100);
144     showForward();
145     insertAtMiddleAfter(30, 60);
146     showForward();
147     insertAtMiddleBefore(30, 70);
148     showForward();
149     insertAtTail(80);
150     showForward();
151     deleteFromHead();
152     showForward();
153     deleteFromTail();
154     showForward();
155     deleteFromMiddle(60);
156     showForward();
157     deleteFromMiddle(70);
158     showForward();
159     while (head != nullptr) {
160         deleteFromHead();
161         showForward();
162     }
163     return 0;
164 }

```