

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  class Node{
4  public:
5      int info;
6      Node *prev;
7      Node *next;
8      Node(int val){
9          info = val;
10         prev = NULL;
11         next = NULL;
12     }
13 };
14 void create(Node **head , int item){
15     Node* newPtr = new Node(item);
16     if(*head == NULL){
17         *head = newPtr;
18     }
19     else{
20         Node* temp = *head;
21         while(temp->next != NULL)temp = temp->next;
22         temp->next = newPtr;
23         newPtr->prev = temp;
24     }
25 }
26 void display(Node **head){
27     cout<<"Displayed data ";
28     Node *temp = *head;
29     while(temp!=NULL){
30         cout<<temp->info<<" ";
31         temp = temp->next;
32     }
33     cout<<endl<<endl;
34 }
35 int isPalindrome(Node **head){
36     Node *last = *head;
37     Node *first = *head;
38     while(last->next != NULL)last = last ->next;
39     while(first != NULL){
40         if((last->info) != (first->info))return 0;
41         first = first->next;
42         last = last->prev;
43     }
44     return 1;
45 }
46
47 void firstInsert(Node **head , int item){
48     Node *newPtr = new Node(item);
49     newPtr->next = *head;
50     if(*head != NULL)(*head)->prev = newPtr;
51     *head = newPtr;
52 }
53 void lastInsert(Node **head , int item){
54     if(*head == NULL){
55         firstInsert(head , item);
56         return;
57     }
58     Node *newPtr = new Node(item);
59     Node *temp = *head;
60     while(temp -> next != NULL)temp = temp->next;
61     temp->next = newPtr;
62     newPtr->prev = temp;
63 }
64 void dataInsert(Node **head , int data , int item){
65     Node *newPtr = new Node(item);
66     Node *temp = *head;

```

```

67     while(temp!= NULL && temp -> info != data)temp = temp->next;
68     if(temp == NULL){
69         cout<<"can't be inserted"<<endl;
70         return;
71     }
72
73     newPtr->next = temp->next;
74     newPtr->prev = temp;
75     if(temp->next != NULL){
76         temp->next->prev = newPtr;
77     }
78     temp->next = newPtr;
79 }
80
81 void firstDelete(Node **head){
82     Node *nextNode = (*head)->next;
83     *head = nextNode;
84     (*head)->prev = NULL;
85 }
86 void dataDelete(Node **head , int data){
87     Node *temp = *head;
88     while(temp!= NULL && temp->info!=data)temp = temp->next;
89     if(temp == NULL){
90         cout<<"cant be deleted"<<endl;
91         return;
92     }
93     Node *prevNode = temp->prev;
94     Node *nextNode = temp->next;
95     prevNode->next = nextNode;
96     if(nextNode != NULL)nextNode->prev = prevNode;
97 }
98 int main(){
99     int n;cin>>n;
100     Node *head = NULL;
101
102     for(int i = 0 ; i < n ; i++){
103         int data;
104         cin>>data;
105         create(&head , data);
106     }
107
108
109     while(1){
110         cout<<"Select operation "<<endl;
111         cout<<"FirstInsert 1"<<endl;
112         cout<<"LastInsert 2"<<endl;
113         cout<<"Data Insert 3"<<endl;
114         cout<<"FirstDelete 4"<<endl;
115         cout<<"Data delete 5"<<endl;
116         cout<<"is palindrome 6"<<endl;
117         int op;cin>>op;
118         if(op == 1){
119             int item;cin>>item;
120             firstInsert(&head , item);
121         }
122         if(op == 2){
123             int item;cin>>item;
124             lastInsert(&head , item);
125         }
126         if(op == 3){
127             int remData , addData;
128             cout<<"remData addData ";
129             cin>>remData>>addData;
130             dataInsert(&head , remData , addData);
131         }
132         if(op == 4){

```

```
133         firstDelete(&head);
134     }
135     if(op == 5){
136         cout<<"Remdata ";
137         int remData;cin>>remData;
138         dataDelete(&head , remData);
139     }
140
141     if(op == 6){
142         if(isPalindrome(&head))cout<<"Palindrome"<<endl;
143         else cout<<"NOT palindrome"<<endl;
144     }
145     else display(&head);
146 }
147
148 firstInsert(&head , 10);
149
150 lastInsert(&head , 10);
151
152 display(&head);
153 cout<<isPalindrome(&head);
154
155 }
```