

코드

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  ////////////////////////////////////////////////// 구조체 init
4  typedef char element;
5  typedef struct ListNode
6  { // 노드 타입을 구조체로 정의한다.
7      element data;
8      struct ListNode* next;
9  }ListNode;
10
11 typedef struct ListType
12 {
13     ListNode* head;
14     int size;
15 }ListType;
16
17
18 void init(ListType* L)
19 {
20     L->head = NULL;
21     L->size = 0;
22 }
23
24 //////////////////////////////////////////////////insert
25 void insertFirst(ListType *L , element e)
26 {
27     ListNode *node = (ListNode*)malloc(sizeof(ListNode));//
28     node->data = e;
29     node->next = L->head;
30     L->head = node;
31     L->size++;
32 }
33
34
35 void insertLast(ListType *L, element e)
36 {
37     ListNode* node = (ListNode*)malloc(sizeof(ListNode));
38     node->data = e;
39     node->next = NULL;
40     if (L->size == 0)
41     {
42         L->head = node;
43     }
44     else {
45         ListNode* p;
46         for (p = L->head; p->next != NULL; p = p->next);
47         p->next = node;
48     }
49     L->size++;
50 }
51
52 void insert(ListType* L, int pos, element e)
53 {
54     if (pos == 1)
55         insertFirst(L, e);
56     else if(pos == L->size + 1)
57         insertLast(L, e);
58     else
59     {
60         ListNode* node = (ListNode*)malloc(sizeof(ListNode));
61         ListNode* p = L->head;
62
63         for (int i = 1; i<pos-1; i++)
64             p = p->next;
65
66         node->data = e;
67         node->next = p->next;
68         p->next = node;
69         L->size++;
70     }
71 }

```

```

119 void deleteFirst(ListType* L)
120 {
121     if (L->size == 0)
122         printf("Empty");
123     else if (L->size == 1)
124     {
125         ListNode* p = L->head;
126         L->head = NULL;
127
128         L->size--;
129     }
130     else
131     {
132         ListNode* p = L->head;
133         L->head = p->next;
134
135         L->size--;
136     }
137 }

```

```

138 void deleteLast(ListType* L)
139 {
140     if (L->size == 0)
141         printf("Empty");
142     else if (L->size == 1)
143     {
144         ListNode* p = L->head;
145         L->head = NULL;
146
147         L->size--;
148     }
149     else
150     {
151         ListNode* p = L->head;
152
153         for (int i = 1; i < L->size - 1; i++)
154             p = p->next;
155
156         p->next = NULL;
157         L->size--;
158     }
159 }

```

```

119 void delete(ListType* L, int pos)
120 {
121     if (pos > L->size)
122         printf("wrong pos");
123     else if (pos == L->size && L->size == 1)
124     {
125         ListNode* p = L->head;
126         L->head = NULL;
127         L->size--;
128     }
129     else if (pos == 1)
130     {
131         ListNode* p = L->head;
132         L->head = p->next;
133
134         L->size--;
135     }
136     else
137     {
138         ListNode* p = L->head;
139
140         for (int i = 1; i < pos - 1; i++)
141             p = p->next;
142
143         p->next = p->next->next;
144         L->size--;
145     }
146 }
147
148
149
150

```

```

151 ////////////////print
152 void print(ListType* L)
153 {
154     ListNode* p;
155     for (p = L->head; p != NULL; p = p->next)
156     {
157         printf("%c => ", p->data);
158     }
159     printf("\n");
160 }
161
162 int main()
163 {
164     ListType L;
165     init(&L);
166
167     insertLast(&L, 'A'); print(&L);
168     insertLast(&L, 'B'); print(&L);
169     insertLast(&L, 'C'); print(&L);
170     insertLast(&L, 'D'); print(&L);
171     insertLast(&L, 'E'); print(&L);
172     insertLast(&L, 'F'); print(&L);
173     insertLast(&L, 'G'); print(&L);
174
175     deleteFirst(&L); print(&L);
176     deleteLast(&L); print(&L);
177     delete(&L, 3); print(&L);
178
179     return 0;
180 }

```

실행결과

```

A =>
A => B =>
A => B => C =>
A => B => C => D =>
A => B => C => D => E =>
A => B => C => D => E => F =>
A => B => C => D => E => F => G =>
B => C => D => E => F => G =>
B => C => D => E => F =>
B => C => E => F =>

```

ListType L = [head] → [A] → [B] → [C] → [D] → [E]

1) deleteFirst(L)

ListNode\* P = L → head : P → next = [B]

L → Head = P → next : [Head] ~~→ [A]~~ ~~→ [B]~~ → [C] ...

2) deleteLast(L)

ListNode\* P = L → head

for (i = 1 → L → size - 1) { P = P → next } P → next = [E]

P → next = Null : [A] → [B] ... [D] ~~→ [E]~~  
↓  
Null

3) delete(L, 3)

ListNode\* P = L → head

for (i = 1 → L → pos - 1) { P = P → next } P → next = [C]

P → next = P → next → next P → next = [D], [A] → [B] ~~→ [C]~~ ~~→ [D]~~ → [E]