

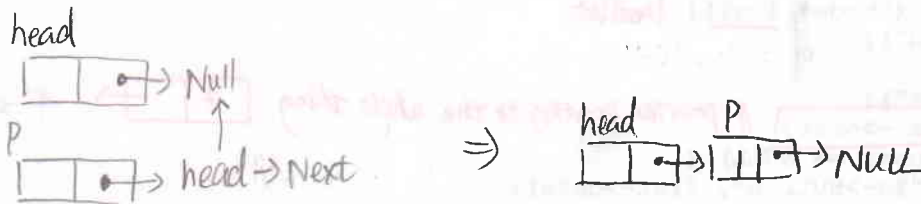
```

1  /*****
2  * File Name: LinkedList-1.c
3  * Author: Alex Yang
4  * Engineering School, NPU
5  * description: Create single linked list by the following
6  *               inputs such as 1 2 3 4, and then
7  *               output will be Head->4->3->2->1->NULL
8  *               and print it out
9  *****/
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 typedef int ElemType;
14
15 typedef struct LNode
16 {
17     ElemType data;
18     struct LNode *next;
19 } LNode; // same as Lnode
20
21 LNode *create_LinkList(void) { create_LinkList() = head
22     int data;
23     LNode *head, *p; // going to point to struct LNode data type
24     head = (LNode *) malloc(sizeof(LNode)); // allocate memory space in heap and casting to LNode data type.
25     head->next = NULL;
26     do {
27         scanf("%d", &data);
28         p = (LNode *) malloc(sizeof(LNode));
29         p->data = data;
30         p->next = head->next; // NULL
31         head->next = p;
32     } while (getchar() != '\n'); // "Enter"
33     return (head);
34 }
35 void printList(LNode* list) { local list
36     printf("Head");
37     while(1) {
38         printf("->");
39         list = list->next; // pointer pointing to the whole thing
40         if(list->next == NULL) {
41             printf("%d->NULL\n", list->data);
42             break;
43         }
44         else
45             printf("%d", list->data);
46     }
47 }
48 int main(void) {
49     LNode *a; // is a pointer of LNode
50     a = create_LinkList(); // return value is a struct LNode datatype pointer
51     printList(a);
52     return 0;
53 }

```



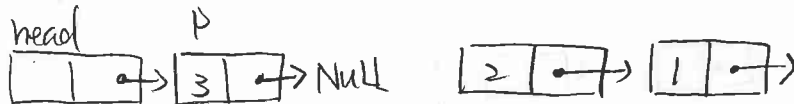
Data = 1 :



Data = 2 :



Data = 3 :



Data = 4 :



Data = "n" :



```

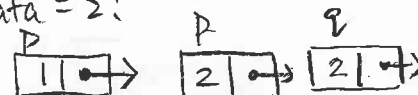
1  /*****
2  * File Name: LinkedList-2.c
3  * Author: Alex Yang
4  * Engineering School, NPU
5  * description: Create single linked list by the following
6  * inputs such as 1 2 3 4, and then
7  * output will be Head->1->2->3->4->NULL
8  * and print it out
9  *****/
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 typedef int ElemType;
14
15 typedef struct Lnode
16 {
17     ElemType data;
18     struct Lnode *next;
19 } LNode;
20
21 LNode *create_LinkList(void) {
22     int data;
23     LNode *head, *p, *q;
24     head = p = (LNode *) malloc(sizeof(LNode));
25     head->next = NULL;
26     do {
27         scanf("%d", &data);
28         q = (LNode *) malloc(sizeof(LNode));
29         q->data = data;
30         q->next = p->next; // = wild pointer.
31         p->next = q;
32         p = q;
33     } while (getchar() != '\n');
34     return (head);
35 }
36 void printList(LNode* list) {
37     printf("Head");
38     while (1) {
39         printf("->");
40         list = list->next;
41         if (list->next == NULL) {
42             printf("%d->NULL\n", list->data);
43             break;
44         }
45         else
46             printf("%d", list->data);
47     }
48 }
49 int main(void) {
50     LNode *a;
51     a = create_LinkList();
52     printList(a);
53     return 0;
54 }

```

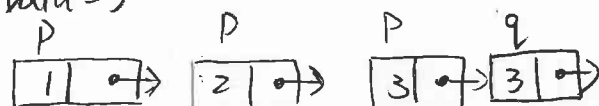
Data = 1:
head



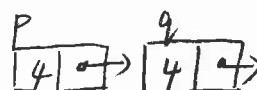
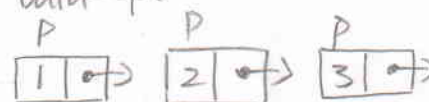
Data = 2:



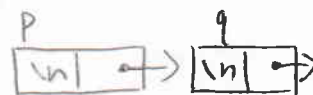
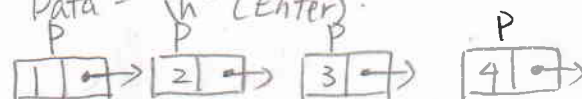
Data = 3:



Data = 4:



Data = "\n" (Enter):



break from loop

head = P



```

1  /*****
2  * File Name: LinkedList-3.c
3  * Author: Alex Yang
4  * Engineering School, NPU
5  * description: Find element in single linked list by index
6  * such as Head->11->12->13->14->NULL, and if
7  * index i = 2, then return value is 12
8  *****/

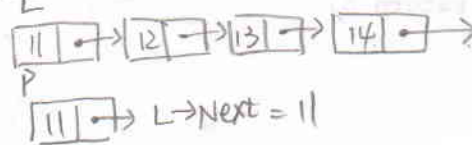
```

```

9  #include <stdio.h>
10 #include <stdlib.h>
11
12 typedef int ElemType;
13
14 typedef struct Lnode
15 {
16     ElemType data;
17     struct Lnode *next;
18 } LNode;
19
20 LNode *create_LinkList(void) {
21     int data;
22     LNode *head, *p, *q;
23     head=p=(LNode *)malloc(sizeof(LNode));
24     head->next=NULL;
25     do{
26         scanf("%d",& data);
27         q= (LNode *)malloc(sizeof(LNode));
28         q->data=data;
29         q->next=p->next;
30         p->next=q;
31         p=q;
32     }while(getchar() != '\n');
33     return (head);
34 }
35
36 ElemType Get_Elem(LNode *L, int i){
37     int j;
38     LNode *p;
39     p=L->next;
40     j=1;
41     while (p!=NULL && j<i){
42         p=p->next;
43         j++;
44     }
45     if(j!=i){
46         printf("Out of range!");
47         exit(0);
48     }
49     else{
50         if(p==NULL){
51             printf("NULL");
52             exit(0);
53         }
54         else
55             return(p->data);
56     }
57 }
58
59 void printList(LNode* list){
60     printf("Head");
61     while(1){
62         printf("->");
63         list = list->next;
64         if(list->next == NULL) {
65             printf("%d->NULL\n", list->data);
66             break;

```

Get_Elem(a,0):



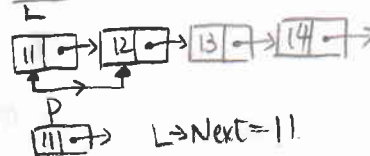
j=1 i=0

while (False) → if statement

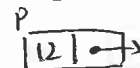
j=1 != i=0

Output: Out of range!

Get_Elem(a,2):



① j=1 i=2



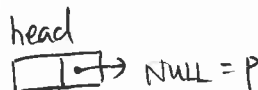
L→Next = 12

j++ ⇒ j=2

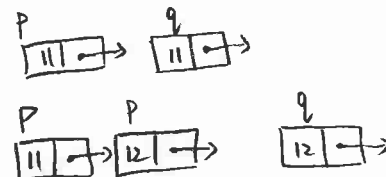
② j=2 i=2

return p→data = 12.

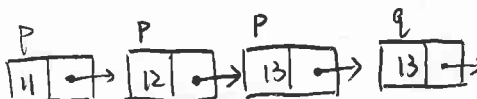
Data=11:



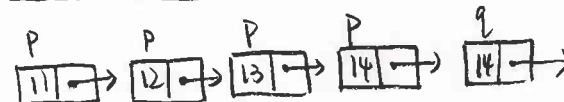
Data=12:



Data=13:



Data=14:



Data="":




↳ break loop

```

1  /*****
2  * File Name: LinkedList-4.c
3  * Author:    Alex Yang
4  *           Engineering School, NPU
5  * description: Check if element is in single linked list
6  *              or not, such as Head->11->12->13->14->NULL,
7  *              element = 13, then print True
8  *****/
9  #include <stdio.h>
10 #include <stdlib.h>
11
12 typedef int ElemType;
13
14 typedef struct LNode
15 {
16     ElemType data;
17     struct LNode *next;
18 } LNode;
19
20 LNode *create_LinkList(void) {
21     int data;
22     LNode *head, *p, *q;
23     head = p = (LNode *) malloc(sizeof(LNode));
24     head->next = NULL;
25     do {
26         scanf("%d", &data);
27         q = (LNode *) malloc(sizeof(LNode));
28         q->data = data;
29         q->next = p->next;
30         p->next = q;
31         p = q;
32     } while (getchar() != '\n');
33     return (head);
34 }
35
36 void Locate_Node(LNode *L, int key) {
37     LNode *p = L->next;
38     while (p != NULL && p->data != key)
39         p = p->next;
40
41     if (p == NULL) printf("It is NOT in the list\n");
42     else if (p->data == key)
43         printf("It is in the list\n");
44 }
45
46 void printList(LNode *list) {
47     printf("Head");
48     while (1) {
49         printf("->");
50         list = list->next;
51         if (list->next == NULL) {
52             printf("%d->NULL\n", list->data);
53             break;
54         }
55         else
56             printf("%d", list->data);
57     }
58 }
59
60 int main(void) {
61     LNode *a;
62     a = create_LinkList();
63     printList(a);
64     int elem = 6;
65     Locate_Node(a, elem);
66     return 0;
67 }

```

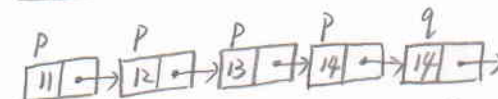
create_LinkList:

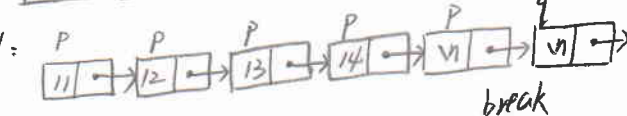
head  $\Leftrightarrow P$

Data = 11: 

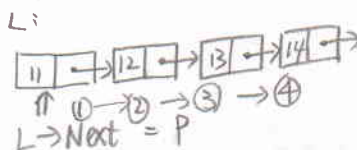
Data = 12: 

Data = 13: 

Data = 14: 

Data = "\n":  break

Locate_Node (a, 6):

L: 

① P->data = 11 P->next = 12

② P->data = 12 P->next = 13

③ P->data = 13 P->next = 14

④ P->data = 14 P->next = NULL

P == NULL

Output: It is NOT in the List.

```

1  /*****
2  * File Name: LinkedList-5.c
3  * Author: Alex Yang
4  * Engineering School, NPU
5  * description: Insert element to single linked list by
6  * index, such as Head->11->12->13->14->NULL
7  * If index=2 and element=345, then new linked
8  * list will be Head->11->12->345->13->14->NULL
9  *****/

```

```

10 #include <stdio.h>
11 #include <stdlib.h>
12
13 typedef int ElemType;
14
15 typedef struct LNode
16 {
17     ElemType data;
18     struct LNode *next;
19 } LNode;

```

```

20
21 LNode *create_LinkList(void) {

```

```

22     int data;
23     LNode *head, *p, *q;
24     head=p=(LNode *)malloc(sizeof(LNode));
25     head->next=NULL;
26     do{
27         scanf("%d",& data);
28         q=(LNode *)malloc(sizeof(LNode));
29         q->data=data;
30         q->next=p->next;
31         p->next=q;
32         p=q;
33     }while(getchar()!='\n');
34     return (head);
35 }

```

```

36
37 void Insert_LNode(LNode *L, int i, ElemType e) {
38     int j=0;
39     LNode *p,*q;
40     p=L->next;
41     while(p!=NULL && j<i-1) {
42         p=p->next;
43         j++;
44     }
45     if(p==NULL || j!=i-1)
46         printf("i is too big OR i is 0!!\n");
47     else{
48         q=(LNode*)malloc(sizeof(LNode));
49         q->data=e;
50         q->next=p->next;
51         p->next=q;
52     }
53 }

```

```

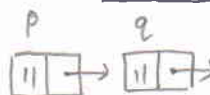
54
55 void printList(LNode* list){
56     printf("Head");
57     while(1){
58         printf("->");
59         list = list->next;
60         if(list->next == NULL) {
61             printf("%d->NULL\n", list->data);
62             break;
63         }
64         else
65             printf("%d",list->data);
66     }

```

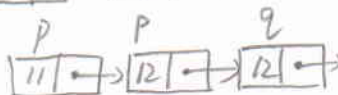
create_LinkList: head



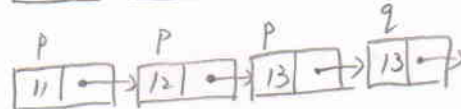
Data=11:



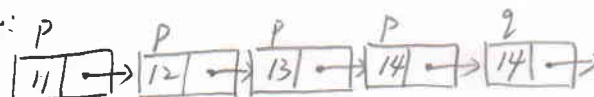
Data=12:



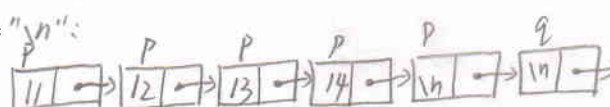
Data=13:



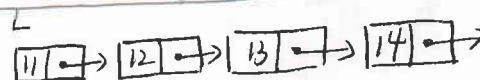
Data=14: p



Data="n": p



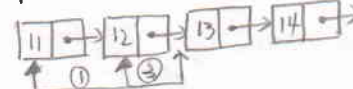
Insert_LNode (a, 2, 12):



L->next

p = L->next j=0 i=2

p



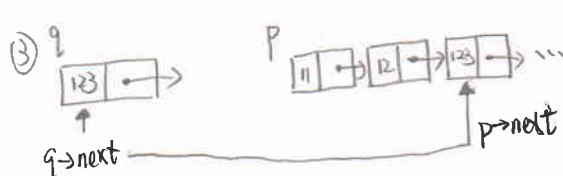
while: ① j=0 i=1

p = p->next = 12.

j=1

② j=1 i=1 break.

else: ③




```

1  /*****
2  * File Name: LinkedList-6.c
3  * Author: Alex Yang
4  * Engineering School, NPU
5  * description: Delete element to single linked list by
6  * index, such as Head->11->12->13->14->NULL
7  * If index=2, and then new linked list will be
8  * Head->11->13->14->NULL
9  *****/

```

```

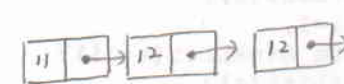
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 typedef int ElemType;
14
15 typedef struct Lnode
16 {
17     ElemType data;
18     struct Lnode *next;
19 } LNode;
20
21 LNode *create_LinkList(void) {
22     int data;
23     LNode *head, *p, *q;
24     head=p=(LNode *)malloc(sizeof(LNode));
25     head->next=NULL;
26     do{
27         scanf("%d",& data);
28         q=(LNode *)malloc(sizeof(LNode));
29         q->data=data;
30         q->next=p->next;
31         p->next=q;
32         p=q;
33     }while(getchar()!='\n');
34     return (head);
35 }
36
37 void Delete_LinkList(LNode *L, int i){
38     int j=1;
39     LNode *p,*q;
40     p=L;
41     q=L->next;
42     while( p->next!=NULL && j<i) {
43         p=q;
44         q=q->next;
45         j++;
46     }
47     if(j!=i){
48         printf("Out of range!\n");
49         exit(0);
50     }
51     else if(q==NULL){
52         printf("i is too big\n");
53         exit(0);
54     }
55     else{
56         p->next=q->next;
57         free(q);
58     }
59 }
60
61 void printList(LNode* list){
62     printf("Head");
63     while(1){
64         printf("->");
65         list = list->next;
66         if(list == NULL){

```


create_LinkList:

head  P

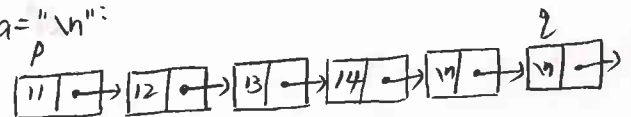
Data=11:  P

Data=12:  P

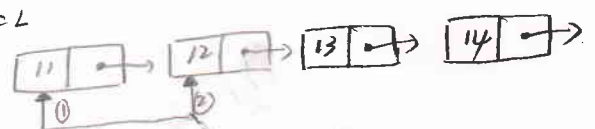
Data=13:  P

Data=14:  P

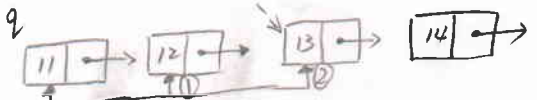
Data="\n":

 P

Delete_LinkList (a,2):

P=L  P

q = L->next:

 P

while: ① P->next=11. j=1 i=2

q=q->next=12 j=2

② P->next=12

q->next=13

break to condition: j=j=2.

③ P->next = q->next = 13.

q ->  (wild pointer)

```

1  /*****
2  * File Name: LinkedList-6.c
3  * Author: Alex Yang
4  * Engineering School, NPU
5  * description: Delete element to single linked list by
6  * key, such as Head->11->12->13->14->NULL
7  * If key=12, then new linked list will be
8  * Head->11->13->14->NULL
9  *****/

```

```

10 #include <stdio.h>
11 #include <stdlib.h>

```

```

12 typedef int ElemType;

```

```

13 typedef struct Lnode
14 {
15     ElemType data;
16     struct Lnode *next;
17 }LNode;

```

```

21 LNode *create_LinkList(void){
22     int data;
23     LNode *head, *p, *q;
24     head=p=(LNode *)malloc(sizeof(LNode));
25     head->next=NULL;
26     do{
27         scanf("%d",& data);
28         q=(LNode *)malloc(sizeof(LNode));
29         q->data=data;
30         q->next=p->next;
31         p->next=q;
32         p=q;
33     }while(getchar()!='\n');
34     return (head);
35 }

```

```

37 void Delete_LinkList(LNode *L, int key){
38     LNode *p=L, *q=L->next;
39     while( q!=NULL && q->data!=key){
40         p=q;
41         q=q->next;
42     }
43     if(q==NULL)
44         printf("Not existing element!!\n");
45     else if(q->data==key){
46         p->next=q->next;
47         free(q);
48     }
49 }

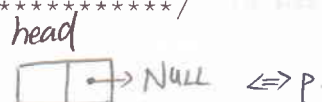
```

```

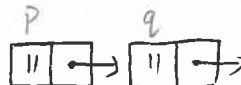
51 void printList(LNode* list){
52     printf("Head");
53     while(1){
54         printf("->");
55         list = list->next;
56         if(list == NULL){
57             printf("NULL");
58             exit(0);
59         }
60         else if(list->next == NULL) {
61             printf("%d->NULL\n", list->data);
62             break;
63         }
64         else
65             printf("%d",list->data);
66     }

```

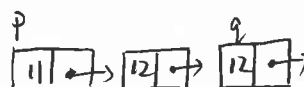
create_Linklist:



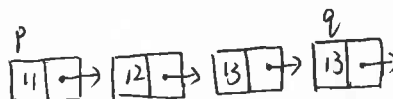
Data=11:



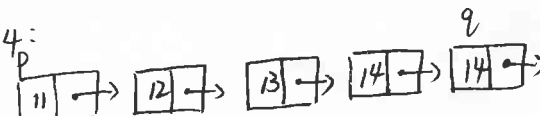
Data=12:



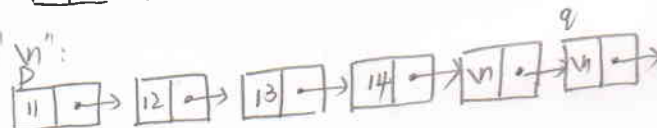
Data=13:



Data=14:

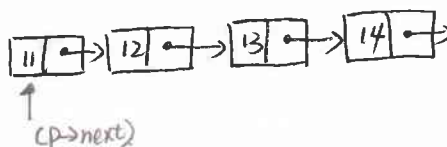


Data="n":

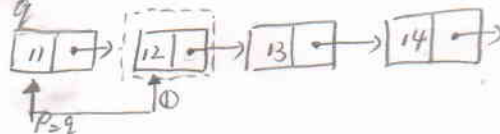


Delete_LinkList (L, 12):

P=L



q=L->next:



while:

①

p=q

q=q->next=12

②

q->data=key break out

condition:

