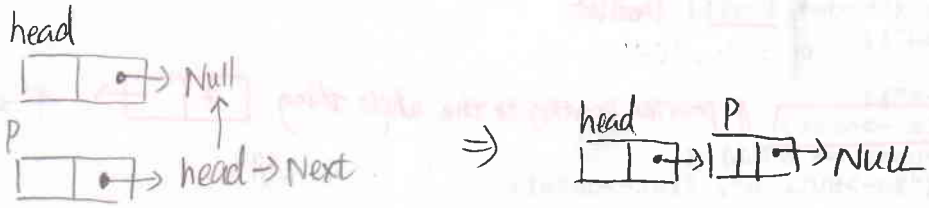
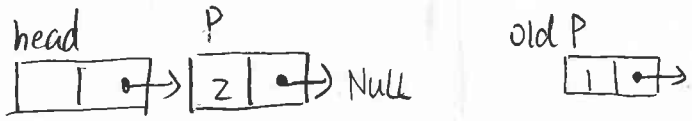
```c
/*********************************************************
 * File Name: LinkedList-1.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Create single linked list by the following
 *              inputs such as 1 2 3 4, and then
 *              output will be Head->4->3->2->1->NULL
 *              and print it out
 *********************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ElemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;  // same as Lnode

LNode  *create_LinkList(void){  create_LinkList() = head
  int data;
  LNode *head, *p; // going to point to struct Lnode data type
  head= (LNode  *) malloc( sizeof(LNode)); // allocate memory space in heap and casting to Lnode data type.
  head->next=NULL;
  do {
    scanf("%d", &data) ;
    p= (LNode  *)malloc(sizeof(LNode));
    p->data = data;
    p->next=head->next; // NULL
    head->next=p ;
  }while(getchar()!='\n'); => "Enter"
  return (head);
}
void printList(LNode* list){  local list
  printf("Head");
  while(1){
    printf("->");
    list = list->next;  // pointer pointing to the whole thing [4|] -> 4->
    if(list->next == NULL) {
      printf("%d->NULL\n", list->data);
      break;
    }
    else
      printf("%d",list->data);
  }
}
int main(void) {
  LNode *a; // is a pointer of Lnode
  a = create_LinkList(); // return value is a struct Lnode datatype pointer
  printList(a);
  return 0;
}
```

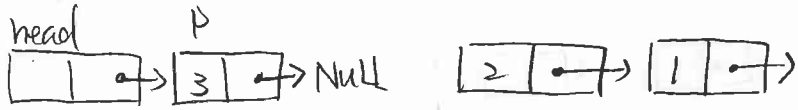
Data = 1 :



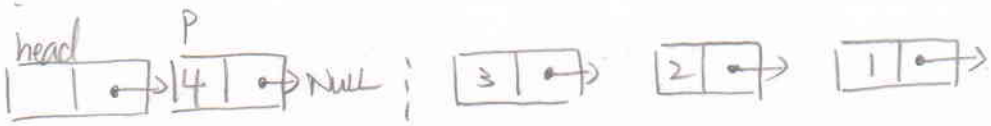
Data = 2 :



Data = 3 :

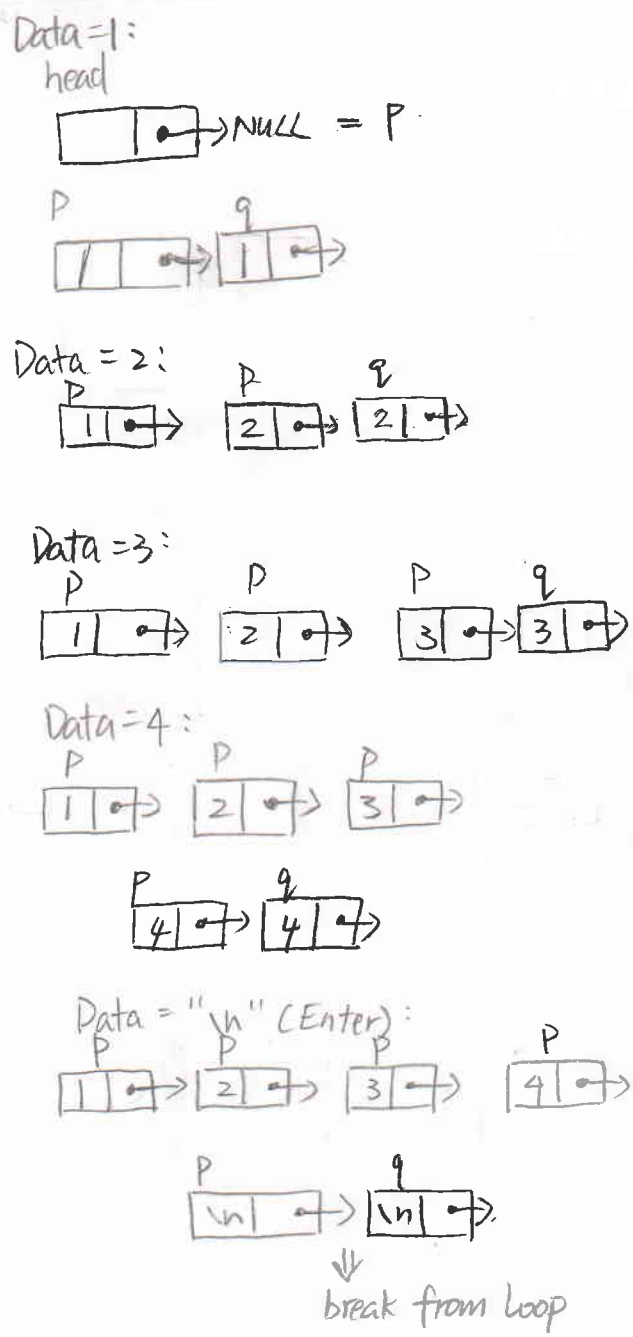


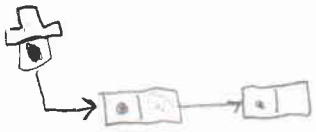
Data = 4 :



Data = "\n" :

```c
/*********************************************************
 * File Name: LinkedList-2.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Create single linked list by the following
 *              inputs such as 1 2 3 4, and then
 *              output will be Head->1->2->3->4->NULL
 *              and print it out
 *********************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ElemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;

LNode  *create_LinkList(void){
  int data;
  LNode *head, *p, *q;
  head=p=(LNode *)malloc(sizeof(LNode));
  head->next=NULL;
  do{
        scanf("%d",& data);
        q= (LNode  *)malloc(sizeof(LNode));
        q->data=data;
        q->next=p->next; //= wild pointer.
        p->next=q;
        p=q ;
  }while(getchar()!='\n');
  return (head);
}
void printList(LNode* list){
  printf("Head");
  while(1){
    printf("->");
    list = list->next;
    if(list->next == NULL) {
      printf("%d->NULL\n", list->data);
      break;
    }
    else
      printf("%d",list->data);
  }
}
int main(void) {
  LNode *a;
  a = create_LinkList();
  printList(a);
  return 0;
}
```

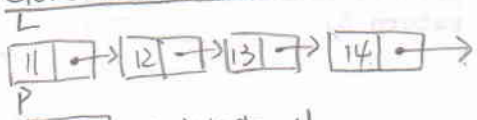




head ≐ P

```c
/*********************************************************
 * File Name: LinkedList-3.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Find element in single linked list by index
 *              such as Head->11->12->13->14->NULL, and if
 *              index i = 2, then return value is 12
 *********************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ÈlemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;

LNode  *create_LinkList(void){
  int data;
  LNode *head, *p, *q;
  head=p=(LNode *)malloc(sizeof(LNode));
  head->next=NULL;
  do{
        scanf("%d",& data);
        q= (LNode  *)malloc(sizeof(LNode));
        q->data=data;
        q->next=p->next;
        p->next=q;
        p=q ;
  }while(getchar()!='\n');
  return (head);
}

ElemType Get_Elem(LNode *L, int  i){
    int j ;
    LNode *p;
    p=L->next;
    j=1;
    while  (p!=NULL && j<i){
        p=p->next;
        j++;
    }
    if(j!=i){
      printf("Out of range!");
      exit(0);
    }
    else{
      if(p==NULL){
        printf("NULL");
        exit(0);
      }
      else
        return(p->data);
    }
}

void printList(LNode* list){
  printf("Head");
  while(1){
    printf("->");
    list = list->next;
    if(list->next == NULL) {
      printf("%d->NULL\n", list->data);
      break;
```

Get_Elem (a.0):



L→Next = 11

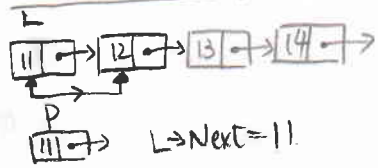
j=1 i=0

while (False) → if statement

j=1 != i=0

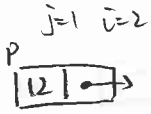
Output: Out of range!

Get_Elem (a,2):



L→Next=11

① j=1 i=2



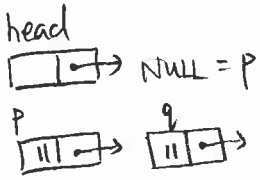
L→Next = 12

j++ ⇒ j=2

② j=2 i=2

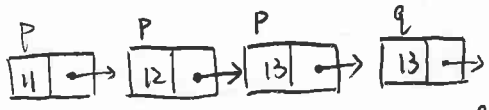
return p→data = 12.

Data=11:

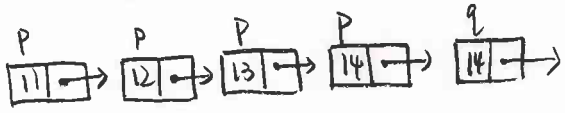


Data=12:

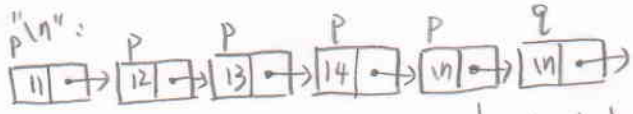
Data=13:



Data=14:



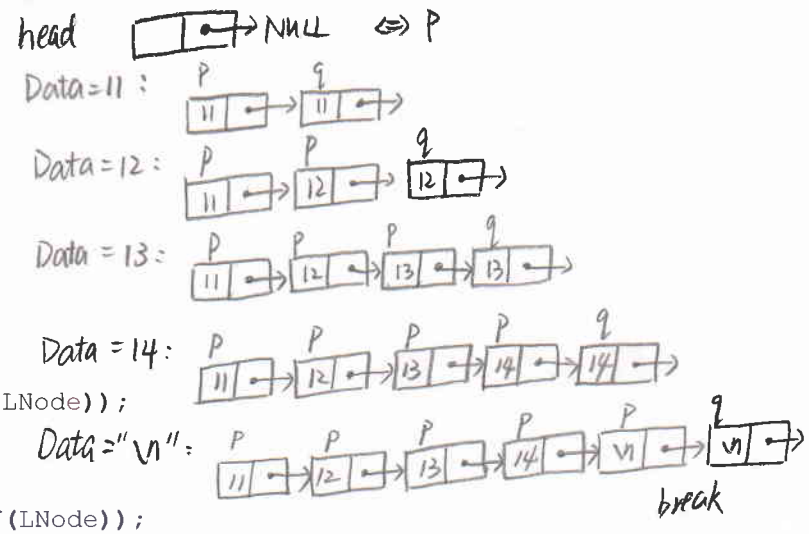
Data = "\n":



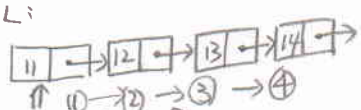
↳ break loop

```c
/*************************************************************
 * File Name: LinkedList-4.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Check if element is in single linked list
 *              or not, such as Head->11->12->13->14->NULL,
 *              element = 13, then print True
 *************************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ÊlemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;

LNode  *create_LinkList(void){
  int data;
  LNode *head, *p, *q;
  head=p=(LNode *)malloc(sizeof(LNode));
  head->next=NULL;
  do{
      scanf("%d",& data);
      q= (LNode  *)malloc(sizeof(LNode));
      q->data=data;
      q->next=p->next;
      p->next=q;
      p=q ;
  }while(getchar()!='\n');
  return (head);
}

void Locate_Node(LNode *L,int key){
    LNode *p=L->next;
    while  ( p!=NULL && p->data!=key)
      p=p->next;

    if(p==NULL) printf("It is NOT in the list\n");
    else if(p->data==key)
      printf("It is in the list\n");
}

void printList(LNode* list){
  printf("Head");
  while(1){
    printf("->");
    list = list->next;
    if(list->next == NULL) {
      printf("%d->NULL\n", list->data);
      break;
    }
    else
      printf("%d",list->data);
  }
}
int main(void) {
  LNode *a;
  a = create_LinkList();
  printList(a);
  int elem=6;
  Locate_Node(a, elem);
  return 0;
}
```

create_LinkList:



Locate_Node(a, 6):



L->Next = P

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

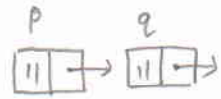
P == NULL

Output: It is NOT in the List

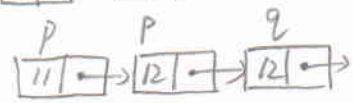
```c
/*************************************************************
 * File Name: LinkedList-5.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Insert element to single linked list by
 *              index, such as Head->11->12->13->14->NULL
 *              If index=2 and element=345, then new linked
 *              list will be Head->11->12->345->13->14->NULL
 *************************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ElemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;

LNode  *create_LinkList(void){
  int data;
  LNode *head, *p, *q;
  head=p=(LNode *)malloc(sizeof(LNode));
  head->next=NULL;
  do{
      scanf("%d",& data);
      q= (LNode  *)malloc(sizeof(LNode));
      q->data=data;
      q->next=p->next;
      p->next=q;
      p=q ;
  }while(getchar()!='\n');
  return (head);
}

void  Insert_LNode(LNode *L, int i, ElemType e) {
    int  j=0;
    LNode *p,*q;
    p=L->next ;
    while(p!=NULL && j<i-1) {
      p=p->next;
      j++;
    }
    if(p==NULL || j!=i-1)
        printf("i is too big OR i is 0!!\n");
    else{
        q=(LNode*)malloc(sizeof(LNode));
        q->data=e;
        q->next=p->next;
        p->next=q;
    }
}

void printList(LNode* list){
  printf("Head");
  while(1){
    printf("->");
    list = list->next;
    if(list->next == NULL) {
      printf("%d->NULL\n", list->data);
      break;
    }
    else
      printf("%d",list->data);
  }
```

Create_LinkList: head

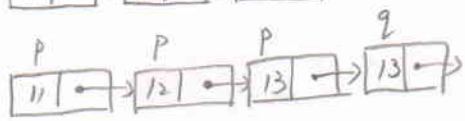
Data=11:



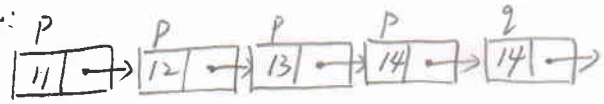
Data=12:



Data=13:

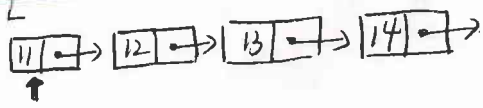


Data=14:



Data="\n":

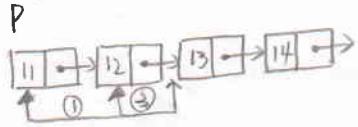
Insert_LNode (a, 2, 123):



L->next

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

p



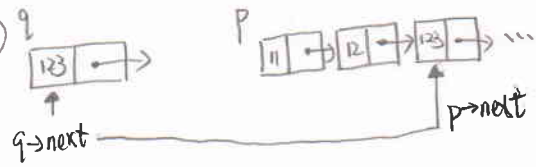
while: ① j=0 i-1=1

p=p->next =12.

j=1

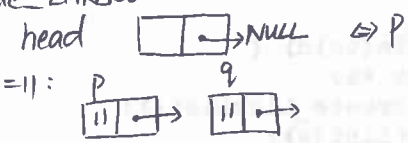
② j=1 i-1=1 break.

↳else: ③



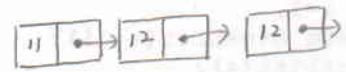
```c
/*************************************************************
 * File Name: LinkedList-6.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Delete element to single linked list by
 *              index, such as Head->11->12->13->14->NULL
 *              If index=2, and then new linked list will be
 *              Head->11->13->14->NULL
 *************************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ElemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;

LNode  *create_LinkList(void){
  int data;
  LNode *head, *p, *q;
  head=p=(LNode *)malloc(sizeof(LNode));
  head->next=NULL;
  do{
      scanf("%d",& data);
      q= (LNode  *)malloc(sizeof(LNode));
      q->data=data;
      q->next=p->next;
      p->next=q;
      p=q ;
  }while(getchar()!='\n');
  return (head);
}

void  Delete_LinkList(LNode *L, int i){
    int  j=1;
    LNode *p,*q;
    p=L;
    q=L->next;
    while( p->next!=NULL && j<i) {
        p=q;
        q=q->next;
        j++;
    }
    if(j!=i){
      printf("Out of range!\n");
      exit(0);
    }
    else if(q==NULL){
        printf("i is too big\n");
        exit(0);
      }
    else{
        p->next=q->next;
        free(q);
    }
}

void printList(LNode* list){
  printf("Head");
  while(1){
    printf("->");
    list = list->next;
    if(list == NULL){
```

create_LinkList:

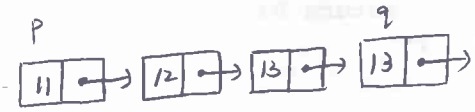


Data=11:

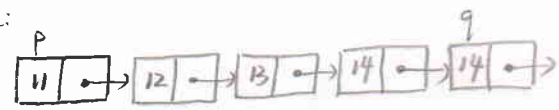
Data=12:



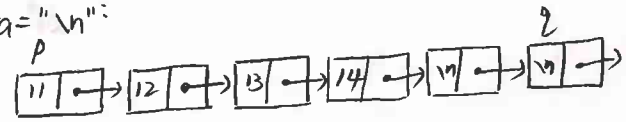
Data=13:



Data=14:

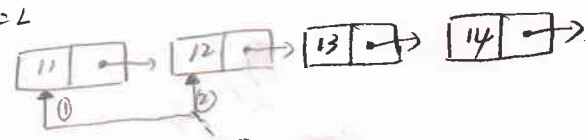


Data="\n":

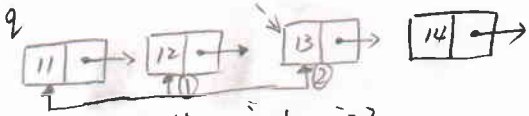


Delete_Linkedlist (a, 2):

p=L



q = L->next :



while : ① p->next = 11 j=1 i=2

q=q->next = 12 j=2

② p->next=12

q->next = 13

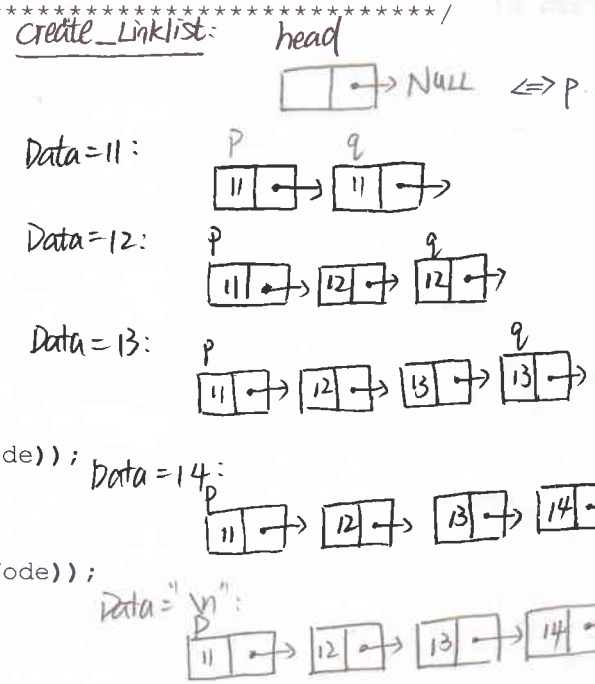
break to condition: j=j=2.

③ p->next = q->next =13.

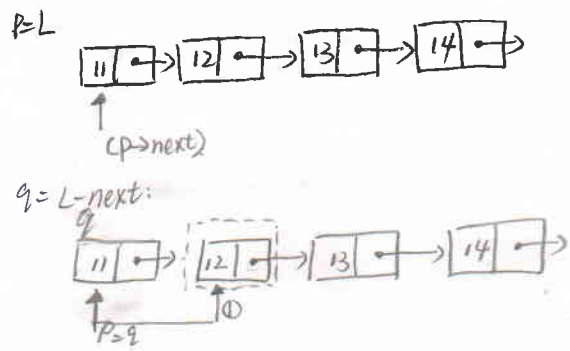
q → (wild pointer)

```c
/*************************************************************
 * File Name: LinkedList-6.c
 * Author:    Alex Yang
 *            Engineering School, NPU
 * description: Delete element to single linked list by
 *              key, such as Head->11->12->13->14->NULL
 *              If key=12, then new linked list will be
 *              Head->11->13->14->NULL
 *************************************************************/
#include <stdio.h>
#include <stdlib.h>

typedef int ElemType;

typedef  struct  Lnode
{
    ElemType  data;
    struct   Lnode  *next;
}LNode;

LNode  *create_LinkList(void){
  int data;
  LNode *head, *p, *q;
  head=p=(LNode *)malloc(sizeof(LNode));
  head->next=NULL;
  do{
      scanf("%d",& data);
      q= (LNode  *)malloc(sizeof(LNode));
      q->data=data;
      q->next=p->next;
      p->next=q;
      p=q ;
  }while(getchar()!='\n');
  return (head);
}

void  Delete_LinkList(LNode *L, int key){
  LNode *p=L, *q=L->next;
  while( q!=NULL && q->data!=key){
    p=q;
    q=q->next;
  }
  if(q==NULL)
    printf("Not existing element!!\n");
  else if(q->data==key){
    p->next=q->next;
    free(q);
  }
}

void printList(LNode* list){
  printf("Head");
  while(1){
    printf("->");
    list = list->next;
    if(list == NULL){
      printf("NULL");
      exit(0);
    }
    else if(list->next == NULL) {
      printf("%d->NULL\n", list->data);
      break;
    }
    else
      printf("%d",list->data);
  }
```

create_Linklist:



Delete_LinkList (L, 12):



while: ① p=q

q=q->next=12

② q->data= key break out

condition: