```cpp
// LinkedListC.cpp : Defines the entry point for the console application.
//

#include <stdlib.h>
#include "LinkList.h"
#include "LinkListClass.h"
#include <stdio.h>

int main()
{

    Node *test_list;
    test_list=creatList();

    for (int i=0;i<10;i++)
    {
        insert(test_list,i+1,i);
    }

    printf("\r\nCreat a list with 0~10 \r\n");
    printList(test_list);

    insert(test_list,5,12);
    printf("\r\nInsert 12 at position 5 \r\n");
    printList(test_list);


    remove(test_list,8);
    printf("\r\nRemove data 8 \r\n");
    printList(test_list);

    makeEmpty(test_list);


    system("pause");
    return 0;
}
```

```
#ifndef LINK_LIST_H
#define LINK_LIST_H

struct Node
{
    int data;
    Node *next;
};

Node* creatList();

void printList(Node *list);

void makeEmpty(Node *list);

bool insert(Node *list, int position, int data);

void remove(Node *list, int data);

bool find(Node *list, int data);

int findKth(Node *list,int position);

#endif
```

```
#include "LinkList.h"
#include <stdio.h>


Node* creatList()
{
    Node *p=new Node;
    p->data=-1;
    p->next=NULL;

    return p;

}
/*==========================================
*
==========================================*/
void printList(Node *list)
{
    printf("=======begin========\r\n");
    while(list)
    {
        printf("%d->",list->data);
        list=list->next;
    }
    printf("null\r\n=======end========\r\n");
}

/*==========================================
*
==========================================*/
void makeEmpty(Node *list)
{
    Node* p=list;

    while(list)
    {
        p=list;

        list=list->next;

        delete p;

    }

}

/*==========================================
*
==========================================*/
bool insert(Node *list, int position, int data)
{
    if (list==NULL)
        return 0;

    while (--position && list)
    {
        list=list->next;
    }

    if (position)
    {
        return 0;
    }
    else
    {
        Node *p=new Node;
        p->data=data;
        p->next=list->next;
        list->next=p;
```

```
        return 1;
    }

}

/*==========================================
*
==========================================*/
void remove(Node *list, int data)
{
    Node *p;

    while (list && list->data!=data)
    {
        p=list;
        list=list->next;
    }

    p->next=list->next;
    delete list;
}

/*==========================================
*
==========================================*/
bool find(Node *list, int data)
{

    while (list && list->data!=data)
    {
        list=list->next;
    }

    return list;
}

/*==========================================
*
==========================================*/
int findKth(Node *list, int position)
{
    while (--position && list)
    {
        list=list->next;
    }

    return list ? list->data : 0;
}
```