<단순연결리스트>

#include <stdio.h>

#include <stdlib.h>

typedef struct ListNode

{

int data;

struct ListNode\* link;

}ListNode;

typedef struct

{

ListNode\*head;

}LinkedListType;

void init(LinkedListType\* L)

{

L-> head = NULL;

}

void addFirst(LinkedListType\* L, int item)

{

ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));

node-> data = item;

node-> link = L-> head;

L->head = node;

}

void add(LinkedListType\* L, int pos, int item)

{

ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));

ListNode\* before = L-> head;

for(int i = 0;i < pos-1; i++)

{

before = before-> link;

}

node -> data = item;

node-> link = before-> link;

before->link = node;

}

void addLast(LinkedListType\*L, int item)

{

ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));

node-> data = item;

node-> link = NULL;

int count = 0;

for(ListNode\*p = L-> head; p!= NULL; p = p->link)

{

count++;

}

ListNode\* q = L-> head;

for(int i = 0;i < count-1; i++)

{

q = q->link;

}

q->link = node;

}

int get(LinkedListType\* L, int pos)

{

ListNode\* p = L-> head;

for(int i = 1; i <pos; i ++)

{

p = p->link;

}

return p->data;

}

void set(LinkedListType\* L, int pos, int item)

{

ListNode\* p = L->head;

for(int i =1; i<pos;i++)

{

p=p->link;

}

p -> data = item;

}

void remove1(LinkedListType\*L, int pos)

{

ListNode\*before = L->head;

for(int i = 0; i< pos-1;i++)

{

before = before->link;

}

ListNode \* next = before;

next = next->link;

next = next->link;

before->link = next;

}

void removeFirst(LinkedListType\*L)

{

ListNode\*p = L->head;

p = p->link;

L->head = p;

}

void removeLast(LinkedListType\*L)

{

int count = 0;

for(ListNode\*p = L-> head; p!= NULL; p = p->link)

{

count++;

}

ListNode\* q = L-> head;

for(int i = 1;i<=count-2; i++)

{

q = q->link;

}

q->link = NULL;

}

void printList(LinkedListType\* L)

{

for(ListNode\* p = L->head; p!=NULL; p=p->link)

{

printf("[%d] -> ", p->data);

}

printf("NULL\n");

}

void main()

{

LinkedListType list;

init(&list);

addFirst(&list,10); printList(&list);

addFirst(&list,20); printList(&list);

addFirst(&list,30); printList(&list);

getchar();

add(&list, 1, 40); printList(&list);

add(&list, 4, 80); printList(&list);

add(&list, 2, 50); printList(&list);

add(&list, 3, 60); printList(&list);

getchar();

addLast(&list, 10); printList(&list);

getchar();

removeFirst(&list); printList(&list);

removeLast(&list); printList(&list);

remove1(&list, 3); printList(&list);

//int pos;

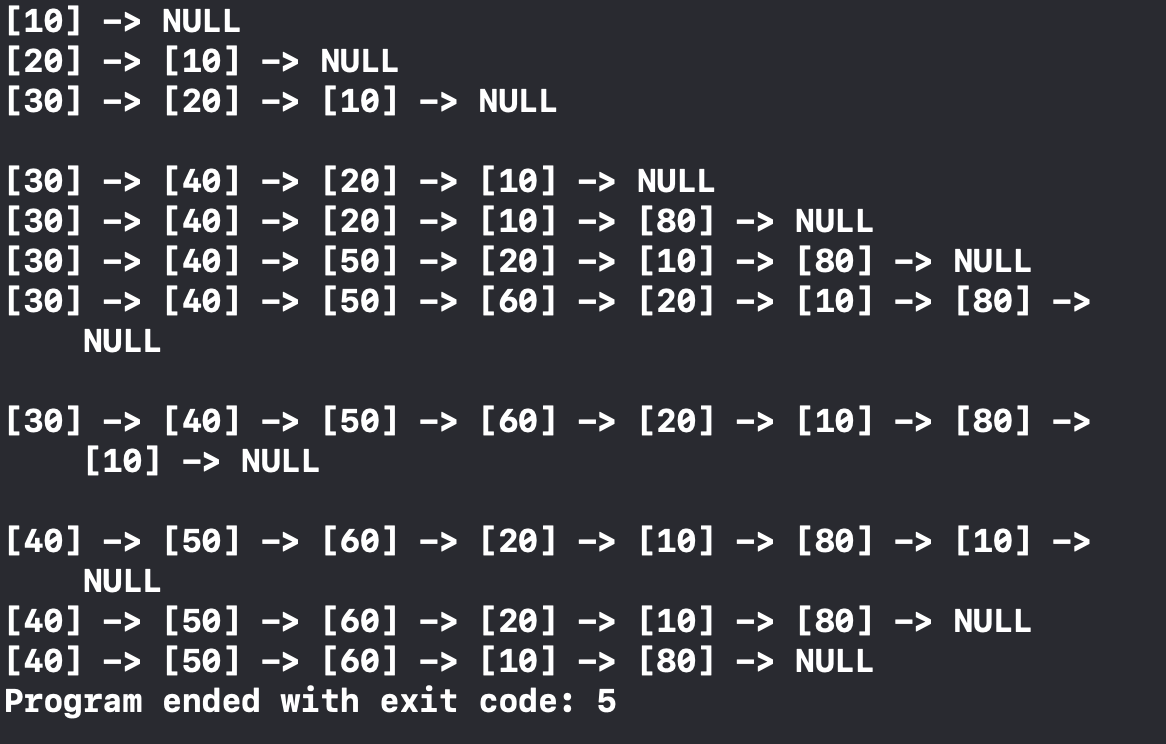
//printf("\n몇 번 노드의 값을 반환할까요? ");

//scanf("%d", &pos);

//printf("%d번 노드의 값은 %d\n", pos, get(&list, pos));

}

결과:



<생일 케이크 배열 ver1>

#include <stdio.h>

#include <stdlib.h>

void buildList(int A[], int n)

{

for(int i = 0; i<n; i++)

{

A[i] = i+1;

}

}

int runSimulation(int A[], int n, int k)

{

int r = 0;

int N = n;

while(n>1)

{

int i = 0;

while(i<k)

{

r = (r+1)%N;

if(A[r] != 0)

i++;

}

A[r] = 0;

n--;

while(A[r] == 0)

{

r = (r+1) % N;

}

}

return A[r];

}

int candle(int A[], int n, int k)

{

buildList(A, n);

return runSimulation(A, n, k);

}

void main()

{

int n,k;

printf("how many candles? ");

scanf("%d", &n);

printf("how many skips? ");

scanf("%d", &k);

int A[n];

printf("%d\n", candle(A, n, k));

}

결과:

텍스트이(가) 표시된 사진

자동 생성된 설명‘

<생일 케이크 배열 ver2>

#include <stdio.h>

#include <stdlib.h>

void buildList(int A[], int n)

{

for(int i = 0; i<n; i++)

{

A[i] = i+1;

}

}

void remove1(int A[], int n, int r)

{

for(int i = r; i<n; i++)

{

A[i] = A[i+1];

}

}

int runSimulation(int A[], int n, int k)

{

int r = 0;

while(n>1)

{

r = (r+k) %n;

remove1(A, n, r);

n--;

}

return A[0];

}

int candle(int A[], int n, int k)

{

buildList(A, n);

return runSimulation(A, n, k);

}

void main()

{

int n,k;

printf("how many candles? ");

scanf("%d", &n);

printf("how many skips? ");

scanf("%d", &k);

int A[n];

printf("%d\n", candle(A, n, k));

}

결과:

텍스트이(가) 표시된 사진

자동 생성된 설명

<생일 케이크 원형연결리스트>

#include <stdio.h>

#include <stdlib.h>

typedef struct ListNode

{

int data;

struct ListNode\* link;

}ListNode;

typedef struct{

ListNode\* head;

}LinkedListType;

void init(LinkedListType\* L)

{

L-> head = NULL;

}

void buildList(LinkedListType\* L, int n)

{

ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));

L-> head = p;

p->data = 1;

for(int i = 2; i<=n; i++)

{

p->link = (ListNode\*)malloc(sizeof(ListNode));

p = p->link;

p-> data = i;

}

p-> link = L->head;

}

int runSimulation(LinkedListType\* L, int n, int k)

{

ListNode\*p = L-> head;

while(p!= p->link)

{

for(int i=1;i<k;i++)

{

p = p->link;

}

ListNode\* pnext = p->link;

p->link = pnext -> link;

free(pnext);

p = p->link;

}

return p->data;

}

int candle(int n, int k)

{

LinkedListType list;

init(&list);

buildList(&list, n);

return runSimulation(&list, n, k);

}

void main()

{

int n,k;

printf("how many candles? ");

scanf("%d", &n);

printf("how many skips? ");

scanf("%d", &k);

printf("%d\n", candle(n, k));

}

텍스트이(가) 표시된 사진

자동 생성된 설명결과: