**<1번>**

#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;

int i;

for (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));

ListNode\* temp;

node->data = item;

if (L->head == NULL)

{

node->link = NULL;

L->head = node;

}

else

{

temp = L->head;

while (temp->link != NULL)

temp = temp->link;

temp->link = node;

node->link = NULL;

}

}

int get(LinkedListType\* L, int pos)

{

ListNode\* p = L->head;

int i;

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

p = p->link;

return p->data;

}

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

{

ListNode\* p = L->head;

int i;

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

p = p->link;

p->data = item;

}

void deleteFirst(LinkedListType\* L)

{

ListNode\* removed;

if (L->head == NULL)

return;

removed = L->head;

L->head = removed->link;

free(removed);

}

void deleteNode(LinkedListType\* L, int pos)

{

int i;

ListNode\* pre = L->head;

ListNode\* removed;

if (pos == 0)

deleteFirst(L);

else

{

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

{

pre = pre->link;

}

removed = pre->link;

pre->link = removed->link;

free(removed);

}

}

void printList(LinkedListType\* L)

{

ListNode\* p;

for (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);

addLast(&list, 80); printList(&list);

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

getchar();

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

add(&list, 1, 50); printList(&list);

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

getchar();

addLast(&list, 25); printList(&list);

deleteFirst(&list); printList(&list);

deleteFirst(&list); printList(&list);

getchar();

deleteNode(&list, 2); printList(&list);

deleteNode(&list, 1); printList(&list);

deleteNode(&list, 0); printList(&list);

deleteNode(&list, 0); printList(&list);

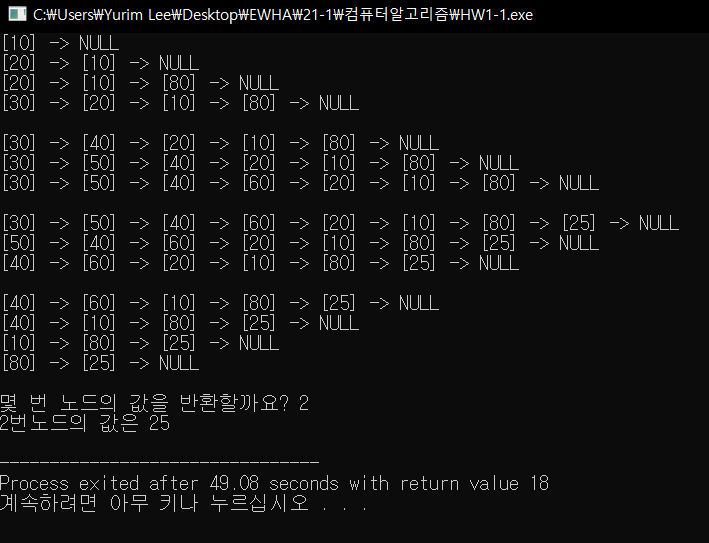
int pos;

printf("\n몇 번 노드의 값을 반환할까요? ");

scanf("%d", &pos);

printf("%d번노드의 값은 %d\n", pos, get(&list, pos));

}



**<2번>**

**[Ver 1]**

#include <stdio.h>

#include <stdlib.h>

void buildList(int A[], int n)

{

int i;

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

{

A[i] = i + 1;

}

return;

}

int runSimulation1(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 -= 1;

while (A[r] == 0) {

r = (r + 1) % N;

}

}

return A[r];

}

int removed(int A[], int n, int pos)

{

int i;

for (i = pos; i < n; i++) {

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

}

return A;

}

int candle(int n, int k)

{

int\* A = malloc(sizeof(int) \* n);

buildList(A, n);

return runSimulation1(A, n, k);

}

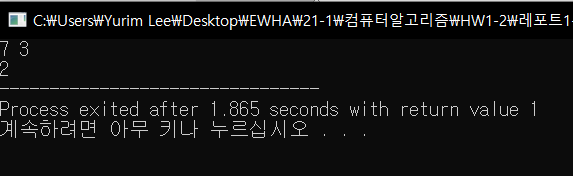
int main(void) {

int n,k;

scanf("%d %d", &n, &k);

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

}



**[Ver2]**

#include <stdio.h>

#include <stdlib.h>

void buildList(int A[], int n)

{

int i;

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

{

A[i] = i + 1;

}

return;

}

int removed(int A[], int n, int pos)

{

int i;

for (i = pos; i < n; i++) {

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

}

return A;

}

int runSimulation2(int A[], int n, int k) {

int r = 0;

while (n > 1) {

r = (r + k) % n;

removed(A,n,r);

n--;

}

return A[0];

}

int candle(int n, int k)

{

int\* A = malloc(sizeof(int) \* n);

buildList(A, n);

return runSimulation2(A, n, k);

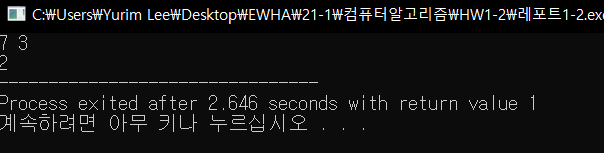
}

int main(void) {

int n,k;

scanf("%d %d", &n, &k);

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

}

**[원형연결리스트 이용]**

#include <stdio.h>

#include <stdlib.h>

typedef struct ListNode

{

int elem;

struct ListNode\* next;

}ListNode;

ListNode\* buildList(int n)

{

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

ListNode\* L;

L = p;

p->elem = 1;

int i;

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

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

p = p->next;

p->elem = i;

}

p->next = L;

return L;

}

int candle(int n, int k)

{

ListNode\* L = buildList(n);

return runSimulation(L, n, k);

}

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

{

ListNode\* p = L;

ListNode\* pnext;

while (p != p->next) {

int i;

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

p = p->next;

}

pnext = p->next;

p->next = p->next->next;

free(pnext);

p=p->next;

}

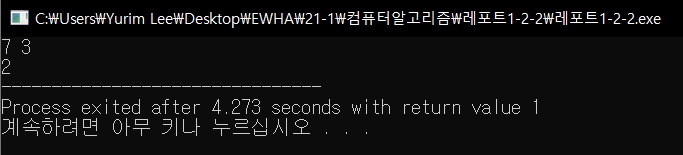
return p->elem;

}

int main(void) {

ListNode\* L;

int n,k;

 scanf("%d %d", &n, &k);

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

}