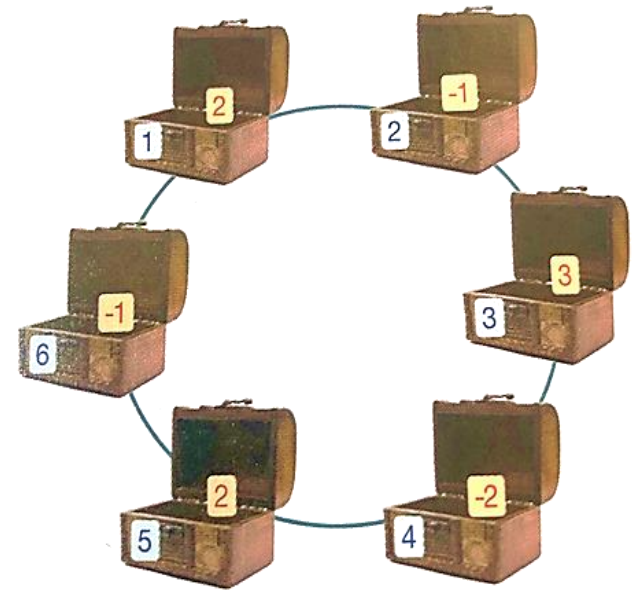


# 보물 상자를 열어라

다음은 보물상자가 여섯 개인 경우를 나타낸 것이다. 첫 번째 여는 보물상자는 항상 1번이다. 1번 보물상자 안에 '2'가 있으니 오른쪽 두 번째 즉, 3번 보물상자를 연다. 그 안에는 '3'이 있으니 오른쪽으로 세 번째 즉, 6번 보물상자를 열어야 한다. 6번 안에는 '-1'이 있으니 왼쪽으로 첫 번째 즉, 5번 보물상자를 연다. 그 안에는 -2가 있으니 오른쪽으로 두 번째 보물상자로 1번인데 이미 열려 있는 보물상자는 제외해야 하므로 다음에 있는 2번 보물상자를 열어야 한다. 2번 안에 -1'이 있으므로 왼쪽으로 첫 번째 보물상자 인데 열려 있는 보물상자를 제외하면 4번 보물상자를 열어야 한다. 즉, 여섯 개 보물상자를 여는 순서는  $1 \rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 4$ 가 된다.



## ❖ 보물상자 여는 순서 출력하기

보물상자 여는 순서를 출력하는 프로그램을 작성하여라

- 입력조건
  - 첫째 줄에는 보물 상자의 개수가 주어진다. ( $1 \leq N \leq 10$ )
  - 둘째 줄에는 보물상자 안에 들어 있는 다음 보물상자 위치를 나타내는 숫자 N개가 주어진다.
- 출력 조건
  - 첫째 줄에 보물상자를 여는 순서를 출력한다.
- 입출력 예시

### 입력 예시

6  
2 -1 3 -2 2 -1



### 출력 예시

1 3 6 5 2 4

## ❖ 이중 원형 연결리스트 구조 만들기

- 이중 원형 연결리스트 구조체 정의

```
#include "stdio.h"
#include "stdlib.h"

// 이중 연결 리스트의 노드 구조를 구조체로 정의
typedef struct node {
    int num;
    int message;
    char check;
    struct node* prev;    // 왼쪽(선행) 노드에 대한 링크
    struct node* next;    // 오른쪽(다음) 노드에 대한 링크
} Node;

// 리스트 위치를 나타내는 tail 노드를 구조체로 정의
typedef struct {
    int countIndex;
    Node* tail;
} List;
```

## ❖ 이중 원형 연결리스트 구조 만들기

- 노드를 생성하고 연결하기(1/2)

```
void insertNode(List* list, int position, int element)
{
    Node* preNode = list->tail;
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->message = element;
    newNode->check='C';

    if (list->countIndex == 0)
    {
        newNode->next = newNode;
        newNode->prev = newNode;
        list->tail = newNode;
    }
}
```

## ❖ 이중 원형 연결리스트 구조 만들기

- 노드를 생성하고 연결하기(2/2)

```
else
{
    for (int i = 0; i < position; i++) {
        preNode = preNode->next;
    }

    newNode->next = preNode->next;
    newNode->prev = preNode;
    newNode->next->prev = newNode;
    preNode->next = newNode;

    if (position == list->countIndex) {
        list->tail = newNode;
    }
}
list->countIndex++;
newNode->num = list->countIndex;
}
```

## ❖ 이중 원형 연결리스트 구조 만들기

### ▪ 공백 리스트 만들기

```
List* createList()
{
    List* list;
    // 테일 노드 할당
    list = (List*)malloc(sizeof(List));
    if (list == NULL) {
        printf( " ERROR\n " );
    }
    else {
        list->tail = NULL;
        list->countIndex = 0;
    }
    return list;
}
```

## ❖ 이중 원형 연결리스트 구조 만들기

- 순서대로 출력하기

```
void printList(List* list)
{
    int i, n;
    Node* node;
    if (list->countIndex == 0) {
        printf("NULL LIST\n");
        return;
    }
    node = list->tail;
    node = node->next;
    n = list->countIndex;
    for (i = 0; i < n; i++)
    {
        printf(" %d, %2d, %c\n",
               node->num, node->message, node->check);
        node = node->next;
    }
}
```

## ❖ 이중 원형 연결리스트 구조 만들기

- 역순으로 출력하기

```
void BackprintList(List* list)
{
    int i, n;
    Node* node;

    if (list->countIndex == 0) {
        printf("NULL LIST\n");
        return;
    }

    node = list->tail;
    n = list->countIndex;
    for (i = 0; i < n; i++)
    {
        printf(" %d, %2d, %c\n",
               node->num, node->message, node->check);
        node = node->prev;
    }
}
```



## ❖ 보물상자 열기

- 보물상자를 여는 순서를 출력

```
void findTreasure(List* list)
{

}
}
```

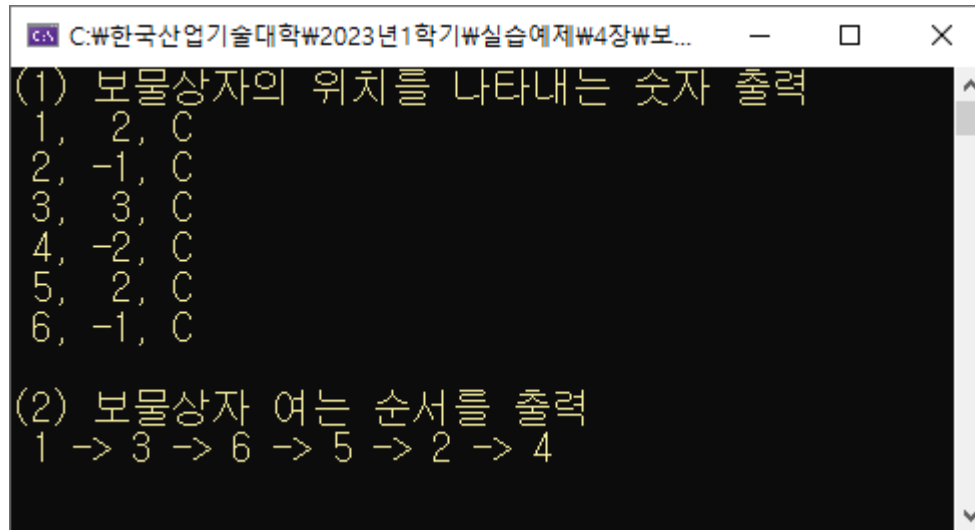
## ❖ 보물상자 열기

- 보물상자의 위치를 나타내는 숫자를 주고 보물상자를 연다.

```
int main()
{
    int k;
    List* list;
    list = createList();
    k = 2; insertNode(list, 0, k);
    k = -1; insertNode(list, 1, k);
    k = 3; insertNode(list, 2, k);
    k = -2; insertNode(list, 3, k);
    k = 2; insertNode(list, 4, k);
    k = -1; insertNode(list, 5, k);
    printf("(1) 보물상자의 위치를 나타내는 숫자 출력");
    getchar();
    printList(list);
    printf("(2) 보물상자 여는 순서를 출력");
    getchar();
    findTreasure(list);
    return 0;
}
```

## ❖ 보물상자 열기

### ■ 실행 결과



```
C:\W한국산업기술대학W2023년1학기W실습예제W4장W보...  
(1) 보물상자의 위치를 나타내는 숫자 출력  
1, 2, C  
2, -1, C  
3, 3, C  
4, -2, C  
5, 2, C  
6, -1, C  
  
(2) 보물상자 여는 순서를 출력  
1 -> 3 -> 6 -> 5 -> 2 -> 4
```

# 해적 널빤지 사형에서 살아남기

크루즈 여행선이 널빤지 사형을 악명이 높은 해적에게 납치되었다. 해적은 납치한 배의 승객을 널빤지에 세워놓고 바다에 빠뜨리다가 마지막까지 남은 한 명만 살려주겠다고 한다. 해적은 납치된 승객 N명을 일렬로 세워놓고 1번부터 순서를 세어 K-1번까지는 뒤로 가서 줄을 서게 하고 K번째 승객은 널빤지로 밀어 바다에 빠뜨린다. 그리고 다시 다음 승객부터 순서를 세어 K-1명은 뒤로 가서 줄을 서게 하고 K번째 승객은 널빤지로 밀어 바다에 푹덩! 다음에 물에 빠질 승객은 누구일까? 마지막까지 살아남으려면 맨 처음 일렬로 줄을 설 때 몇 번째 자리에 있어야 할까?



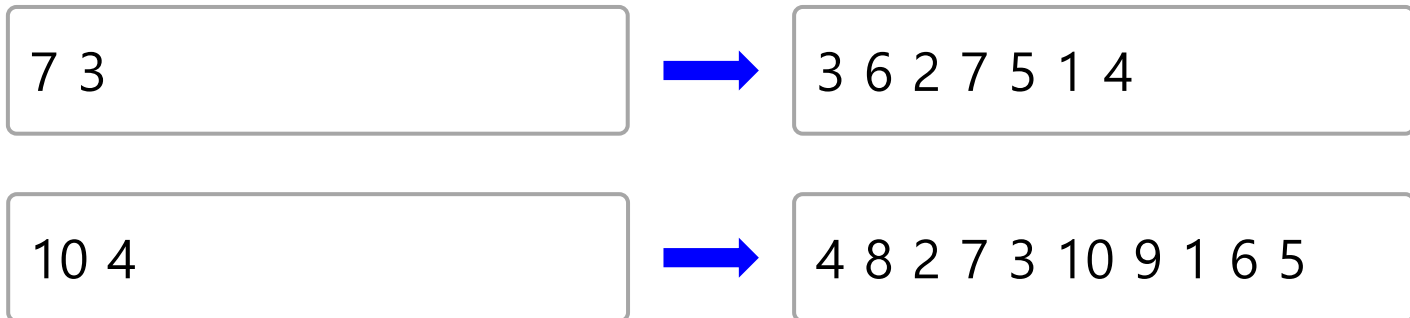
승객이 일곱 명일 경우에 세 번째 승객을 널빤지로 보낸다면 순서는 3, 6, 2, 7, 5, 1, 4가 된다. 네 번째 자리에 있으면 마지막 순서가 되어 살아남게 된다. 이 때 N은 7이 되고 K는 3이 된다.

## ❖ 문제 : 순서 출력하기

N명 중 K번째 사람을 널빤지로 보내는 경우의 순서를 출력하는 프로그램을 작성하라.

- ① 입력조건
  - 첫째 줄에 N과 K에 대한 정수가 주어진다.  
( $1 \leq N \leq 50$ ,  $1 \leq K \leq N$ )
- ② 출력조건
  - 첫째 줄에 널빤지로 선택되는 자리 번호를  
순서대로 출력한다.

### ③ 입출력 예시



## ❖ 문제 해결

### ① 설계

$n$ 과  $k$ 가 자연수일 때  $k < n$ 이라고 가정한다.  $n$ 명이 동그랗게 모여 있을 때 임의의 한 명부터 순서를 세어  $k$ 번째 사람을 모임에서 제외한다. 남은  $n-1$ 명에서 다시 다음 사람부터 순서를 세어  $k$ 번째 사람을 모임에서 제외한다. 이것을 아무도 남지 않을 때까지 계속해서 반복한다. 이때 모임에서 제외되는 사람의 순서를  $(n, k)$ 요세푸스 순열이라고 하며 마지막으로 제외되는 사람을 구하는 문제를 요세푸스 문제라고 한다. 예를들어 요세푸스 순열은  $[3, 6, 2, 7, 5, 1, 4]$ 이며 네번째에 위치한 사람이 마지막으로 제외된다.  $n$ 과  $k$ 의 관계식을 구하면 다음과 같다.

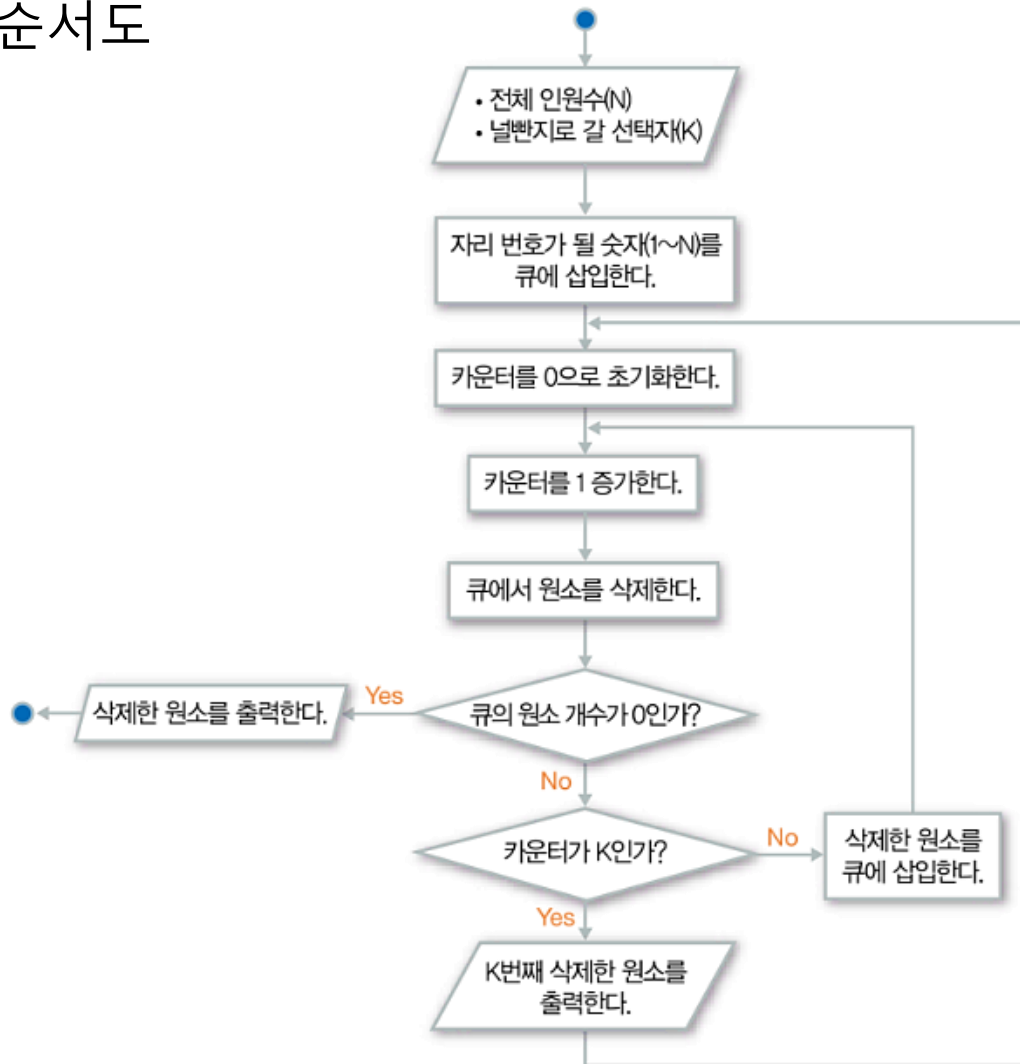
$$f(n, k) = (f(n-1, k) + k - 1) \bmod n + 1$$

큐를 사용하면 위의 수학적 관계식보다 간단히 해결할 수 있다.

- ㉠ 앞에서 부터  $K-1$ 번 까지를 순서대로 뒤로 보내야 하므로 큐에서 삭제하여 다시 큐에 삽입한다.
- ㉡  $K$ 번은 큐에서 삭제하여 출력한다.
- ㉢ ㉠~㉡를 반복하다가 큐의 마지막 원소를 삭제하여 출력하면 작업이 완성된다.

## ❖ 문제 해결

### ② 순서도



## ❖ 널빤지 살아남기 문제 해결 코드

```
#include "stdio.h"
#include "stdlib.h"
#define cQ_SIZE 50

typedef int element; // 큐 원소(element)의 자료형을 int로 정의

typedef struct {
    element queue[cQ_SIZE]; // 1차원 배열 큐 선언
    int front, rear;
} QueueType;

// 공백 원형 큐의 생성
QueueType* createQueue()
{
    QueueType* cQ;
    cQ = (QueueType*)malloc(sizeof(QueueType));
    cQ->front = 0; // front 초기값 설정
    cQ->rear = 0; // rear 초기값 설정
    return cQ;
}
```



## ❖ 널빤지 살아남기 문제 해결 코드

```
// 원형 큐가 공백 상태인지 검사하는 연산
int isEmpty(QueueType* cQ)
{
    if (cQ->front == cQ->rear) {
        printf(" Circular Queue is empty! ");
        return 1;
    }
    else
        return 0;
}

// 원형 큐가 포화 상태인지 검사하는 연산
int isFull(QueueType* cQ)
{
    if (((cQ->rear + 1) % cQ_SIZE) == cQ->front) {
        printf(" Circular Queue is full! ");
        return 1;
    }
    else
        return 0;
}
```

## ❖ 널빤지 살아남기 문제 해결 코드

```
// 원형 큐의 rear에 원소를 삽입하는 연산
void enqueue(QueueType* cQ, element item)
{
    if (isFull(cQ))
        return;
    else {
        cQ->rear = (cQ->rear + 1) % cQ_SIZE;
        cQ->queue[cQ->rear] = item;
    }
}

// 원형 큐의 front에서 원소를 삭제하고 반환하는 연산
element dequeue(QueueType* cQ)
{
    if (isEmpty(cQ))
        exit(1);
    else {
        cQ->front = (cQ->front + 1) % cQ_SIZE;
        return cQ->queue[cQ->front];
    }
}
```

## ❖ 널빤지 살아남기 문제 해결 코드

```
void survive(QueueType* cQ, int K)
{

}

}
```

## ❖ 널빤지 살아남기 문제 해결 코드

```
int main()
{
    QueueType* cQ = createQueue(); // 큐 생성
    element i, N, K;
    element cnt = 1;
    printf("두개의 정수를 입력하세요 : ");
    scanf_s("%d %d", &N, &K);

    printf("\n ** 널빤지 사형에서 살아남기 **\n\n");
    for (i = 0; i < N; i++) {
        enqueue(Q1, i + 1);
    }

    survive(cQ, K);

    getchar();
    getchar();
    return 0;
}
```

## ❖ 널빤지 살아남기 실행 결과

```
C:\> C:\한국산업기술대학\2023년1학기\실습예제\6장\널빤지...  
두개의 정수를 입력하세요 : 7 3  
  
** 널빤지 사형에서 살아남기 **  
  
3 6 2 7 5 1 4
```

```
C:\> C:\한국산업기술대학\2023년1학기\실습예제\6장\널빤지...  
두개의 정수를 입력하세요 : 10 4  
  
** 널빤지 사형에서 살아남기 **  
  
4 8 2 7 3 10 9 1 6 5
```