

## 〈자료구조 실습〉 - C언어 복습

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

**[ 문제 1 ]** 하나의 양의 정수 X를 입력 받아 다음 수식의 결과를 출력하는 프로그램을 작성하시오.

$$1 + (1+2) + (1+2+3) + (1+2+3+4) + \dots + (1+2+\dots+X)$$

입력 예시

출력 예시

4	20
---	----

다음의 함수를 작성하여 사용하시오..

- sum( ) 함수
  - 인자: int형 변수 n
  - 반환값: 1부터 n까지의 합을 int형으로 반환
- main( ) 함수
  - 입출력 수행
  - sum( ) 함수를 반복 호출하여 결과 값 계산

**[ 문제 2 ]** 10개의 정수를 입력으로 받아, 가장 큰 수부터 내림차순으로 정렬하여 출력하는 프로그램을 작성 하시오.

입력 예시 1

출력 예시 1

1 3 5 7 9 2 4 6 8 10	□10 9 8 7 6 5 4 3 2 1
----------------------	-----------------------

입력 예시 2

출력 예시 2

13 56 27 89 43 76 32 68 91 8	□91 89 76 68 56 43 32 27 13 8
------------------------------	-------------------------------

내림차순은 다음의 방법으로 수행하시오.

- ABC( ) 함수
  - 인자: int 배열, 정수 k
  - 배열의 k 번째 정수부터 마지막 정수(즉, 9번째 정수)중 가장 큰 정수를 찾고, 이를 k 번째 위치의 정수와 교환한다.
  - 반환 값: 없음

- main( ) 함수는 다음과 같이 수행
  - 입력되는 정수를 int 형 배열에 저장
  - 배열과 k를 인수로 하여, 함수 ABC( )의 호출을 9번 반복한다. 매 호출 시, k의 값은 0, 1, 2, .... 으로 변한다.
  - 배열에 저장된 정수를 순서대로 출력

**[ 문제 3 ]** N개의 정수를 입력 받아 배열에 저장한 후 ( $N \leq 50$ ), 다시 배열에서 값을 교환하고 싶은 인덱스 a 와 b를 입력 받는다. 인덱스 a 와 b 위치의 요소 값들이 교환된 배열을 출력하는 프로그램을 작성하시오.

- 인덱스 a, b의 상대적 크기는 정해져 있지 않다. 즉,  $a < b$  혹은  $a > b$ .

입력 예시 1

출력 예시 1

6        □ N 3 2 0 1 4 6 2 4       □ a b	□ 3 2 4 1 0 6
--	---------------

다음 함수를 작성하여 사용하시오.

- swap( ) 함수
  - 인자: 두 개의 int **포인터** (배열 원소 값이나 인덱스가 인자가 아님에 유의)
  - 인자가 가리키는 두 변수의 값 교환
  - 반환 값: 없음
- main( ) 함수
  - 입출력 수행
  - swap( ) 함수를 호출하여 두 개의 값 교환.

**[ 문제 4 ]** 사용자로부터 공백을 포함하지 않는 문자열을 하나 입력 받아 예제와 같이 왼쪽으로 한 칸씩 shift하여 출력하는 프로그램을 작성 하시오.

- 입력 받는 문자열 길이는 최대 100 이다.
- 출력 시 반복문을 사용하지 않고 문자열 출력(%)을 사용하시오. 문자 출력(%c) 사용금지

입력 예시 1

출력 예시 1

abcde	abcde bcdea cdeab deabc eabcd
-------	---

**[ 문제 5 ]** 사용자로부터 두 개의 시각을 입력 받아서 두 시각 사이의 차이를 계산하여 출력하는 프로그램을 작성하시오.

- 시각은 시, 분, 초로 구성되는 구조체로 정의하라.
- 두 번째 시각이 첫 번째 시각보다 항상 늦은 시각이라고 가정한다.
- 시간차가 없는 경우에 분과 초만 출력하는 것이 아니라 시 분 초, 0 10 20 으로 출력한다.

입력 예시 1

출력 예시 1

10 20 30    □    10시 20분 30초	1 44 40
12 05 10    □    12시 05분 10초	

입력 예시 2

출력 예시 2

1 10 20    □    1시 10분 20초	2 10 10
3 20 30    □    3시 20분 30초	

**[ 문제 6 ]** 5명 학생의 이름과 기말고사 점수를 입력 받아, 평균 이하의 점수를 받은 학생의 이름을 출력하는 프로그램을 작성하시오.

- 학생의 이름은 공백 없이 최대 9개 영어 문자이다.
- 1명의 학생의 정보 (이름과 점수)를 저장하는 구조체를 정의하고, 5명의 학생의 정보는 구조체 배열에 저장하시오.

입력 예시 1

출력 예시 1

akim 75 bkim 85 ckim 65 dkim 95 ekim 100	akim ckim
--	--------------

**[ 문제 7 ]** N개의 정수를 두 번 입력받아, 예시와 같이 역방향으로 더하여 출력하는 프로그램을 작성하시오.

입력 예시 1

출력 예시 1

3    □ 배열 크기 N=3 1 2 3 5 10 15	□ 16 12 8    □ 16=1+15, 12=2+10, 8=3+5
--------------------------------------	--

입력 예시 2

출력 예시 2

4    □ 배열 크기 4 3 8 9 5 0 1 5 6	□ 9 3 10 5
--------------------------------------	------------

※ **주의:** 문제 명세에서 N의 크기에 제한을 두지 않았음에 주의하라.  
(도움말: 필요한 배열은 동적할당 받아 사용하여야 함)

**[ 문제 8 ]** N명의 학생에 대한 정보(이름, 국어 성적, 영어 성적, 수학 성적)를 입력 받아, 각 학생의 평균 성적과 GREAT 혹은 BAD을 출력하는 프로그램을 작성하시오.

- 평균은 소수 첫째 자리까지 출력
- GREAT 혹은 BAD는 다음 기준을 적용하여 출력:
  - 국어, 영어, 수학 성적 중 어느 한 과목이라도 90 이상일 경우 GREAT 출력
  - 국어, 영어, 수학 성적 중 어느 한 과목이라도 70 미만일 경우 BAD 출력
  - GREAT 여부를 BAD 여부보다 먼저 출력하며, GREAT과 BAD 모두 출력될 경우 공백으로 구분

입력 예시 1

출력 예시 1

2 Kim 100 82 34 Young 90 100 99	Kim 72.0 GREAT BAD Young 96.3 GREAT
---------------------------------------	--

프로그램은 다음 조건을 만족시켜야 한다.

- 다음 멤버를 가지는 student 구조체를 정의하여 사용하시오.
  - 이름: 길이가 1 이상 7 이하인 공백을 포함하지 않는 문자열
  - 국어 성적, 영어 성적, 수학 성적: 각각 정수형 변수. 성적은 0 이상 100 이하인 정수
  - 평균 성적: 실수형 변수
- N명의 학생 정보는 구조체 배열을 동적으로 할당하여 저장하시오.

※ **도움말:** 이 문제도 N의 크기에 제한을 두지 않았다. 따라서 필요한 배열은 동적할당 받아 사용하여야 한다.

- 정적할당을 사용하면, 그 보다 큰 입력은 처리할 수 없다. 예를 들어, 크기가 100인 구조체 배열을 선언하여 구현하면, 그 프로그램은 100보다 큰 N에 대해 처리할 수 없다.

## 〈자료구조 실습〉 - 알고리즘 분석

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↪ 이 후는 각 입력과 출력에 대한 설명이다.

※ **참고:** 이번 주의 주요 실습 내용은 알고리즘의 성능을 비교하는 [문제 3-2]이다. 아래 **A, B**는 [문제 3-2]를 해결하기 위해 필요한 두 가지 참고 사항이다.

### A. 난수발생 함수

- 프로그램에 따라서는 사용자 개입없이 한 개 또는 여러 개의 난수를 공급받아야 제대로 작동하는 경우가 있다. 아주 간단한 예로 사람과 번갈아 가며 주사위를 던지는 프로그램의 경우를 들 수 있다. 컴퓨터가 주사위를 던질 차례에서 사람이 대신 던져줄 수는 없다. 사용자 개입없이 컴퓨터 스스로 무작위 수를 생성해야만 하는 것이다. 아래는 이런 상황에서 사용할 난수발생 함수에 대한 도움말이다.
- 난수란 주사위 눈수처럼 특정한 나열 순서나 규칙이 없이 생성된 무작위 수를 말하며 영어로는 random number라고 한다. 그리고 난수발생 함수란 난수를 자동생성시켜 공급해주는 함수를 말한다.
- C 언어에서는 시스템 라이브러리를 통해 난수발생 함수를 제공한다. 함수 **rand()**가 **0 ~ RAND\_MAX** 범위의 무작위 정수를 반환한다. 여기서 **RAND\_MAX**는 **rand()**가 반환할 수 있는 최대수며 이 값은 시스템마다 다를 수 있다.
- 난수발생 함수를 사용하기 위해서는 헤더파일 **stdlib.h**가 필요하다. 즉, 코드 상단에 **#include <stdlib.h>**를 써줘야 **rand()**를 사용할 수 있다. 이 헤더파일에 **rand()**의 원형이 포함되어 있으며 **rand()**가 발생시킬 수 있는 최대수인 **RAND\_MAX**도 정의되어 있다.
- 사용 예 1

```
#include<stdio.h>
#include<stdlib.h>

int main(void) {
    printf("%d\n", rand());
    return 0;
}
```
- 하지만 위 코드를 여러 번 실행시켜 보면 계속 같은 난수가 나오는 것을 볼 수 있다.

무작위 수의 연속이 아니라 같은 수의 연속이며, 이는 주사위를 아무리 던져도 같은 수가 나오는 상황이라 제대로 된 난수라고 할 수도 없다.

- 이 문제를 해결하기 위해, 즉 매번 다른 난수를 발생시키기 위해 시드(seed)값을 설정하는 방법이 있다. 시드 값을 달리 하면 함수 **rand()**에서 발생시키는 난수가 매번 달라진다.
- 시드 값을 설정하기 위해 사용하는 함수가 바로 **srand()**이다. **srand()**는 이를 호출할 때 전달하는 인자를 기반으로 하여 난수를 초기화시키는 역할을 한다.
- **srand()**의 인자로써 함수 **time()**을 권장한다. **time()**은 인자로 **NULL**을 전달하면 1970년 1월 1일 0시 (UTC 타임존) 이후 현재까지 흐른 초 수를 반환한다. 시간은 멈추지 않고 계속해서 흐르므로 **time()** 함수가 반환한 현재의 초 수를 인자로 하여 **srand()**를 호출하면 난수 기준 값이 (무작위라 할 수 있는) 현재 초 수로 초기화되는 것이다. 따라서 **srand(time(NULL))**과 같이 호출하면 된다. 잊지말 것은, **time()**을 사용하기 위해서는 상단에 **#include <time.h>**를 추가해야 하는 것이다. 정리하면, 최종적인 코드는 아래와 같다.

#### ○ 사용 예 2

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main(void) {
    srand(time(NULL));
    printf("%d\n", rand());
    return 0;
}
```

#### ○ Tip

##### **rand() % n**

- 0 ~ (n - 1) 범위의 난수를 생성한다.
- 예 1: 0 ~ 1 사이의 정수 난수를 발생시키고 싶다면, **rand() % 2**로 호출.
- 예 2: 0 ~ 10000 사이의 정수 난수를 발생시키고 싶다면, **rand() % 10001**로 호출.

##### **rand() % n + m**

- m ~ (n - 1 + m) 범위의 난수를 생성한다.
- 예 1: 1 ~ 6 사이의 정수 난수(예: 주사위 눈수)를 발생시키고 싶다면, **rand() % 6 + 1**로 호출.
- 예 2: 1000 ~ 9999 사이의 정수 난수를 발생시키고 싶다면, **rand() % 9000 + 1000**으로 호출.

`(((((long) rand() << 15) | rand()) % 1000000) + 1`

- 1 ~ 1,000,000 범위의 난수를 생성한다.

- **RAND\_MAX = 32,767**인 경우 이 수보다 큰 범위의 난수를 발생시키기 위한 방법:  
**32,767**의 16진수 표현은 0X7FFF, 2진수 표현은 0111 1111 1111 1111이므로 **rand()** 호출은 15비트로 표현할 수 있는 난수를 반환한다. 이 값을 왼쪽으로 15비트만큼 shift해준 뒤 두번째 **rand()** 호출로부터 얻은 난수와 **OR** 연산을 해주면 30비트로 표현할 수 있는 난수를 얻을 수 있다. 이 난수의 범위는 0 ~ 1,073,741,823이며 이 값을 1,000,000으로 나머지 연산을 한 뒤 1을 더해주면 1 ~ 1,000,000 범위의 난수를 얻게 된다.

## B. 시간측정 함수

- 알고리즘의 실행시간을 측정하는 데는 점근적 분석 방법과 실제 시간 측정, 두 가지가 있다.
- 점근적 분석에 의한 시간 측정은 big-Oh 값을 구하는 이론적 방식을 말하며, 실제 시간 측정은 알고리즘 실행에 소요되는 cputime을 측정하는 것을 말한다.
- 점근적 방식에 의한 측정은 <자료구조 및 실습> 교재 1장에서 배운대로 이론적 방식에 의해 측정할 수 있으며, 실제 시간 측정은 라이브러리 함수를 이용하여 어떤 알고리즘 실행에 소요되는 실제 cputime을 측정한다. 아래는 라이브러리 함수를 이용한 실제 시간 측정에 관한 도움말이다.
- 라이브러리 함수 가운데 일반적인 시간측정 함수인 **clock()**을 사용하면 시간이 정밀하게 나오지 않는 문제가 발생한다. 대안으로 **QueryPerformanceCounter()** 함수를 사용하면 정밀한 시간을 출력할 수 있다. 구체적인 사용 방법은 다음과 같다.
  - 헤더파일로 **windows.h**를 추가한 후,
  - **LARGE\_INTEGER** 변수 선언하고,
  - **QueryPerformanceFrequency()** 함수를 통해 타이머의 주파수를 변수에 저장한 후,
  - 시간을 측정하고 싶은 작업의 전후에 **QueryPerformanceCounter()**를 호출하고 그 반환값들을 이용하여 계산, 출력하면 된다.
- 사용 예

```
#include <stdio.h>
#include <Windows.h>

int main(void){
    LARGE_INTEGER ticksPerSec;
    LARGE_INTEGER start, end, diff;

    QueryPerformanceFrequency(&ticksPerSec);
    QueryPerformanceCounter(&start);
    // 시간을 측정하고 싶은 작업(예: 함수 호출)을 이곳에 삽입
    QueryPerformanceCounter(&end);

    // 측정값으로부터 실행시간 계산
    diff.QuadPart = end.QuadPart - start.QuadPart;
    printf("time: %.12f sec\n\n", ((double)diff.QuadPart)/((double)ticksPerSec.QuadPart));
    return 0;
}
```



- 작동 원리: 메인보드에 고해상도의 타이머가 존재하는데 이를 이용하여 특정 실행 시점들의 CPU 클럭수들을 얻어온 후 그 차이를 이용하여 작업 시간을 구한다. **clock()** 함수와 달리 **1us** 이하의 시간까지 측정한다.

**QueryPerformanceFrequency()** : 타이머의 주파수(초당 진동수)를 얻는 함수

**QueryPerformanceCounter()** : 타이머의 CPU 클럭수를 얻는 함수

작업 전후의 클럭수 차를 주파수로 나누면 작업 시간(초, **sec**)을 구할 수 있고, **ms**단위로 출력하기 위해선 결과 값에 **1,000**을 곱해주면 된다.

- **clock()** 함수와 비교: **clock()**은 초당 **1,000**번의 측정을 통해 **1ms**의 시간을 측정할 수 있는데 비해, **QueryPerformanceCounter()**는 초당 **10,000,000**번의 측정으로 **0.1us**의 시간까지 측정할 수 있다. 초당 클럭수는 **time.h**를 헤더로 추가한 후 **CLOCKS\_PER\_SEC**을 출력하여 알 수 있고, 타이머의 주파수는 **QueryPerformanceFrequency()**를 통해 알 수 있다.

[ 문제 1 ] 나머지 연산

'%(modulo) 연산자는 나눗셈의 나머지를 반환한다. 덧셈과 뺄셈 연산자만을 사용하여 **a**를 **b**로 나눈 나머지를 반환하는 **modulo(a, b)** 함수와 이를 테스트할 프로그램을 작성하시오. 단, **a ≥ 0**, **b > 0** 인 정수다.

※ **힌트: modulo(a, b)** 함수는 **O(a/b)** 시간에 실행하도록 작성할 수 있다.

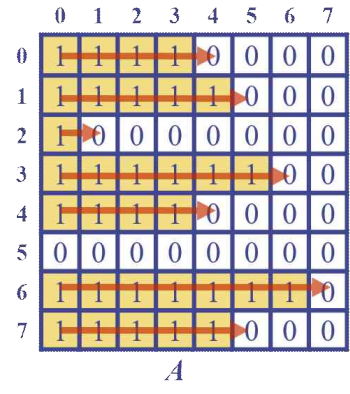
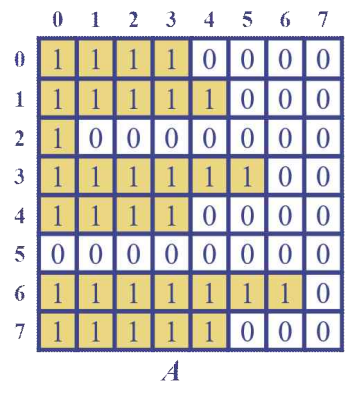
입력 예시 1	출력 예시 1
14 3      ↳ a=14, b=3	2            ↳ 14 % 3
입력 예시 2	출력 예시 2
3 3        ↳ a=3, b=3	0            ↳ 3 % 3
입력 예시 3	출력 예시 3
0 4        ↳ a=0, b=4	0            ↳ 0 % 4

필요 함수:

- **modulo()** 함수
  - 인자: 정수 **a**, **b**
  - 반환값: **a % b**

[ 문제 2 ] 비트행렬에서 최대 1행 찾기

$n \times n$  비트 행렬 **A**의 각 행은 1과 0으로만 구성되며, **A**의 어느 행에서나 1들은 해당 행의 0들보다 앞서 나온다고 가정하자. 행렬 **A**를 입력받아, 가장 많은 1을 포함하는 행을 찾는 프로그램을 작성하시오. 그러한 행이 여러 개 있을 경우 그 가운데 가장 작은 행 번호를 찾아야 한다.

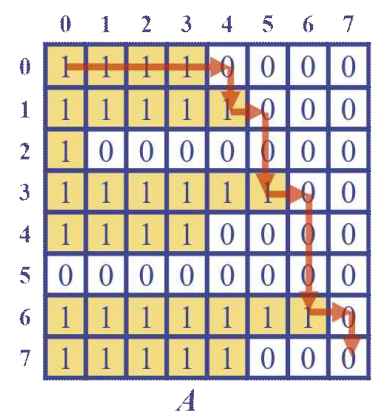


- 한다.
- 예:  $8 \times 8$  비트 행렬 **A** : 6행이 가장 많은 1을 포함한다(아래 왼쪽 그림 참고).
  - 참고로, 아래 의사코드 함수 **mostOnesSlowVersion**은 1이 가장 많은 행을 찾기는 하지만, 실행시간이  $O(n)$ 이 아니라  $O(n^2)$ , 즉 2차 시간(quadratic time)이다(위 오른쪽 그림 참고).

```
Alg mostOnes(A, n)                                {slow version}
  input bit matrix A[n × n]
  output the row of A with most 1's

1. row, jmax ← 0
2. for i ← 0 to n - 1
   j ← 0
   while ((j < n) & (A[i, j] = 1))
     j ← j + 1
   if (j > jmax)
     row ← i
     jmax ← j
3. return row                                     {Total  $O(n^2)$ }
```

- 위 함수보다 빠른 해결은 다음과 같다(오른쪽 그림 참고).
  - 행렬의 좌상 셀에서 출발한다.
  - 0이 발견될 때까지 행렬을 가로질러 간다.
  - 1이 발견될 때까지 행렬을 내려간다.
  - 마지막 행 또는 열을 만날 때까지 위 2, 3 단계를 반복한다.
  - 1을 가장 많이 가진 행은 가로지른 마지막 행이다.



- 빠른 버전 해결은 최대  $2n$ 회의 비교를 수행하므로, 명백히  $O(n)$ -시간, 즉 선형 시간(linear time) 알고리즘이다.

입출력 형식:

1) 입력: **main** 함수는 다음 값들을 표준입력받는다.

- 첫 번째 라인: 정수  $n$  ( $n \times n$  행렬에서  $n$  값, 단  $n \leq 100$ 으로 전제함)
- 두 번째 이후 라인:  $n \times n$  비트 행렬 원소들(행우선 순서)

2) 출력: **main** 함수는 1이 가장 많은 행 번호를 출력한다. 단, 첫 번째 행 번호는 0이다.

입력 예시 1

8	↳ $n = 8$ ( $8 \times 8$ 행렬)
1 1 1 1 0 0 0 0	↳ 각 비트는 공백으로 구분
1 1 1 1 1 0 0 0	
1 0 0 0 0 0 0 0	
1 1 1 1 1 1 0 0	
1 1 1 1 0 0 0 0	
0 0 0 0 0 0 0 0	
1 1 1 1 1 1 1 0	
1 1 1 1 1 0 0 0	

출력 예시 2

6 ↳ 1이 가장 많은 행 번호: 6

필요 함수:

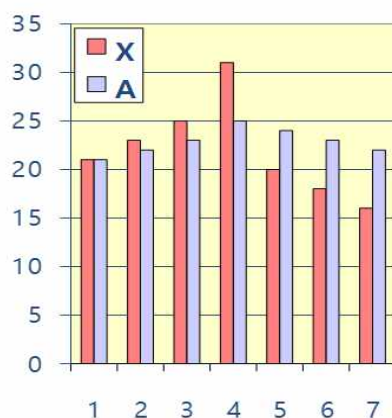
- **mostOnes(A, n)** 함수
  - 인자: 비트 행렬  $A$ , 정수  $n \leq 100$  ( $A$ 의 크기)
  - 반환값: 정수 (최대 1 행 번호)
  - 시간 성능:  $O(n)$

### [ 문제 3 ] 누적 평균

원시 데이터값들로 구성된 배열  $X$ 의  $i$ -번째 **누적평균**(prefix average)이란  $X$ 의  $i$ -번째에 이르기까지의  $(i + 1)$ 개 원소들의 평균이다. 즉,

$$A[i] = (X[0] + X[1] + \dots + X[i]) / (i + 1)$$

누적평균은 경제, 통계 분야에서 오르내림 변동을 순화시킴으로써 대략적 추세를 얻어내기 위해 사용된다. 일례로 부동산, 주식, 펀드 등에도 자주 활용된다.. 이 문제는 배열  $X$ 의 누적평균(prefix average) 배열  $A$ 를 구하는 프로그램을 구현하고 테스트하는데 관한 것이다.



- 아래 의사코드 함수 **prefixAverages1**은 위 정의를 있는 그대로 이용하여 누적평균값들을 2차 시간에 구한다.

※ **참고:** 각 명령문 오른 편 중괄호 내의 수식은 실행시간 분석을 위한 근거로서, 해당 명령문이 수행하는 치환, 반환, 산술 및 비교 연산 등 기본 명령들의 수행 횟수를 나타낸다.

```

Alg prefixAverages1(X, n)      {ver.1}
  input array X of n integers
  output array A of prefix averages of X

1. for i ← 0 to n - 1          {n}
    sum ← 0                     {n}
    for j ← 0 to i             {1 + 2 + ... + n}
        sum ← sum + X[j]       {1 + 2 + ... + n}
    A[i] ← sum/(i + 1)          {n}
2. return A                    {1}
                                {Total O(n2)}
```

- 위의 의사코드 함수 **prefixAverages1**의 내용을 살펴보면, **i** 번째 외부 반복에서는, 바로 전 **i - 1**번째 반복에서 구했던 **[0 ~ i - 1]**의 합에, **i + 1** 번째 원소 값 한 개만을 더해 현재 합을 얻어 평균을 구한다. 따라서 이를 수정하여 이전의 **i - 1**번째까지의 합을 보관하여 다음 반복으로 전달하는 방식으로 반복한다면 현재 합을 구하는데 필요한 시간을 단축할 수 있다는 것을 알 수 있다. 이렇게 중간 합을 보관하는 방식으로 알고리즘을 개선한 함수 **prefixAverage2**는 누적평균값들을 선형 시간에 구할 수 있게 된다.

**문제 3-1>** 함수 **prefixAverages1**과 **prefixAverages2**, 그리고 이들을 테스트할 수 있는 **main** 함수를 구현하여 아래 테스트를 수행하라.

입출력 형식:

- 1) **main** 함수는 아래 형식의 표준입력을 사용하여 배열 **X**를 초기화한 후 각 함수를 호출한다.

입력 : **main** 함수는 다음 값들을 표준입력 받는다.

첫 번째 라인: 정수 **n** (배열 **X**의 크기)

두 번째 이후 라인: **X[0] X[1] X[2] ...** (배열 **X**, 한 라인 상의 양의 정수 수열)

※ **힌트:** **n**의 크기에는 제한이 없다. 따라서 동적 할당을 사용하여야 함)

- 2) **main** 함수는 아래 형식의 표준출력을 사용하여 각 함수로부터 반환된 배열 **A**를 출력한다.

출력 : **A[0] A[1] A[2] ...**

(배열 **X**와 같은 크기의 배열 **A**의 원소들을 나타내는 한 라인 상의 양의 정수

수열로서 첫 번째 라인은 **prefixAverages1**의 출력을, 두 번째 라인은

**prefixAverages2**의 출력을 나타낸다)

- 3) 평균 계산 시 소수점 이하를 반올림하여 정수로 구한다. 정확한 반올림을 위해, %.f를 쓰지 말고 int 성질을 이용해서 반올림하라.

입력 예시 1

3	↳ 배열 X 크기
5 1 9	↳ 배열 X

출력 예시 1

5 3 5	↳ prefixAverages1의 출력
5 3 5	↳ prefixAverages2의 출력

입력 예시 2

6	↳ 배열 X 크기
1 3 2 10 6 8	↳ 배열 X

출력 예시 2

1 2 2 4 4 5	↳ prefixAverages1의 출력
1 2 2 4 4 5	↳ prefixAverages2의 출력

문제 3-2> 위 **main** 함수를 수정하여 아래 절차로 두 함수 **prefixAverage1**과 **prefixAverage2** 각각의 실행시간을 측정 비교하라.

※ 주의:

- 1) **main** 함수는 배열 **X**의 크기 **n**을 표준입력받는다.
- 2) **main** 함수는 **난수발생 함수**를 사용하여 크기 **n**의 **1~10,000** 사이의 정수 배열 **X**를 초기화한 후 각 함수를 한 번씩 호출한다.
- 3) **main** 함수는 각 함수로부터 반환 직후 해당 함수의 실행시간 데이터를 표준출력한다(배열 초기화 시간을 포함하지 않도록 주의).
- 4) 예를 들어 실행시간 = 0 이 되지 않도록, 그리고 두 함수의 성능 비교가 가능하도록 소수점 정밀도를 높여야 한다. 사용 컴퓨터의 성능 문제 때문에 피치 못할 경우에만 아래 입력 예시와는 다른 배열 크기 값들을 사용하여 실행시간 데이터를 얻어도 좋다.

입력 예시 1

100000	↳ 배열 X 크기
--------	-----------

출력 예시 1

0.421289721ms	↳ prefixAverages1의 cpu time
0.054142322ms	↳ prefixAverages2의 cpu time

입력 예시 2

200000	↳ 배열 X 크기
--------	-----------

출력 예시 1

0.852323142ms	↳ prefixAverages1의 cpu time
0.054142322ms	↳ prefixAverages2의 cpu time

입력 예시 1

500000	↳ 배열 X 크기
--------	-----------

출력 예시 1

0.421289721ms	↳ prefixAverages1의 cpu time
0.054142322ms	↳ prefixAverages2의 cpu time

필요 함수:

- **prefixAverages1(X, n), prefixAverages2(X, n)** 함수
  - **prefixAverage1**: 느린 버전
  - **prefixAverage2**: 빠른 버전
  - 인자: 정수 배열 **X** (원시 데이터값들), 정수 **n** (배열 **X**의 크기)
  - 반환값: 정수 배열 **A** (누적평균값들)

※ 참고: 문제 3-2의 실습 목적

- 본 문제는 실행시간(cpu time)을 측정하여 비교분석하는 것이 목적인 실습으로, 출력결과는 컴퓨터마다, 실행할 때마다 다르게 나올 수 있다.
- 코딩평가시스템(OJ 시스템)으로 자동 채점되지 않는 문제로, 출력 결과(실행시간)를 통해 두 함수의 실행시간이 차이가 나는지,  $x$ 가 증가함에 따라 실행시간이 어떤 비율로 증가하는지를 확인해보자.
- 위의 입력 예시 **1, 2, 3**을 그대로 사용한 결과를 출력하라.

## 〈자료구조 실습〉 - 재귀

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

**[ 문제 1 ]** 양의 정수 N을 입력 받아, 1부터 N까지의 합을 구하는 프로그램을 작성하시오.

입력 예시 1

출력 예시 1

10 ↳ 입력 정수 N	55
--------------	----

- 다음 재귀 관계를 이용하시오. (재귀 함수 사용).
- 1부터 N까지의 합  $\Rightarrow$  1부터 (N-1)까지의 합에 N을 더한 값

**[ 문제 2 ]** 양의 정수를 입력 받아, 각 자리의 수를 높은 자릿수부터 차례로 출력하는 프로그램을 작성하시오.

입력 예시 1

출력 예시 1

3408 ↳ 입력 정수	3 4 0 8
--------------	------------------

- int 범위의 정수가 입력된다고 가정하고, %d를 이용하여 입력받아 저장할 것 (%c 또는 %s 사용 금지)
  - 다음 예에서 보여주는 재귀 관계를 이용하시오. (재귀 함수 사용).
  - 예1: 3408의 자릿수 출력  $\Rightarrow$  340의 자리수를 출력하고, 일의 자릿수 8 출력
  - 예2: 1234567의 자릿수 출력  $\Rightarrow$  123456의 자리수를 출력하고, 일의 자릿수 7 출력
- 도움말: 3408이 주어졌을 때, 340과 8을 구하기 위해 어떤 연산자를 사용해야 할지 생각해보자.

**[ 문제 3 ]** 양의 정수를 입력 받아, 각 자리의 수를 낮은 자릿수부터 차례로 출력하는 프로그램을 작성하시오. (나머지 조건은 문제2와 동일)

입력 예시 1

출력 예시 1

3408 ↳ 입력 정수	8 0 4 3
--------------	------------------

도움말: 출력순서에 차이가 있음에 주목하고, 이 차이점이 재귀 함수에 어떻게 반영되는 지를 파악하라.

**[ 문제 4 ]** N 개의 정수를 입력 받아 ( $N \leq 20$ ), 최댓값을 구하는 프로그램을 작성하시오.

입력 예시 1

출력 예시 1

5	↪ N = 5	8	↪ 최댓값
4 1 8 3 7	↪ 입력 정수		

- 다음 예에서 보여주는 재귀 관계를 이용하여 구현하시오 (재귀 함수 사용).
  - 예: (4, 1, 8, 3, 7)의 최댓값  $\Rightarrow$  (4, 1, 8, 3)의 최댓값과 7 중 큰 값

**[ 문제 5 ]** 원반의 개수 N을 입력받아, 하노이 탑 문제의 수행과정을 예시와 같이 출력하는 프로그램을 작성하시오.

- 하노이 탑(towers of Hanoi) 문제
  - 세 개의 말뚝: A, B, C
  - 초기 상황: 직경이 다른  $N > 0$  개의 원반이 A에 쌓여 있음
  - 목표: 모든 원반을 A로부터 C로 옮김
- 조건
  - 한 번에 말뚝의 가장 위에 있는 한 개의 원반만 이동 가능
  - 직경이 큰 원반은 작은 원반 위에 놓일 수 없음
  - 남은 말뚝을 보조 말뚝으로 사용 가능

입력 예시 1

출력 예시 1

2	↪ 원반 개수 N = 2	A B	↪ 말뚝 A의 맨 위 원반을 말뚝 B로 이동
		A C	↪ 이하 동일
		B C	

- 하노이 탑의 재귀 관계는 교재 또는 인터넷 자료 참고하고, 재귀 함수로 구현하시오.

**[ 문제 6 ]** 두 개의 양의 정수를 입력받아, 이 두 정수의 최대공약수(gcd)를 유클리드 호제법으로 계산하여 출력하는 프로그램을 작성하시오.

입력 예시 1

출력 예시 1

12 8	↪ 두 양의 정수	4	↪ 12와 8의 최대공약수
------	-----------	---	----------------

- 유클리드 호제법을 재귀함수를 사용하여 구현하시오. (다음 예시와 반복버전 코드를 참고)

```
gcd(12, 8)
= gcd(8, 12%8) = gcd(8,4)
= gcd(4, 8%4) = gcd(4,0)
```

한 수가 0이 되었을 때,  
다른 수가 최대공약수

// a와 b의 최대공약수 (반복버전)

```
while (b > 0){
    r = a % b;
    a = b;
    b = r;
}
```

while문 종료 후 a의 값이 최대 공약수



**【 문제 7 】** 공백 없는 영어 문자열 하나를 입력받아, 특정 문자가 몇 번 나타나는 지 검사하는 프로그램을 작성하시오.

- 문자열의 길이는 최대 100이고, 문자 검사 시 대소문자를 구별한다.

입력 예시 1

출력 예시 1

SheIsAStudent s	↳ 문자열 ↳ 검사할 문자	1	↳ 소문자 s가 한 번 나타남
--------------------	-------------------	---	------------------

- 다음 재귀 관계를 이용하여 구현하시오 (재귀 함수 사용).
  - 어떤 문자열에서 's' 의 개수  $\Rightarrow$  첫 번째 문자가 's' 인지의 여부와 두 번째 문자 이후에서 나타나는 's'의 개수를 이용하여 계산

## 〈자료구조 실습〉 - 배열

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

**[ 문제 1 ]**  $N$  ( $3 \leq N \leq 100$ ) 개의 정수로 이루어진 수열  $X$ 를 “뒤집기 정보”에 의해 변환한 최종 결과를 출력하는 프로그램을 작성하시오. “뒤집기” 방식은 다음과 같다. 예를 들어, 10개 정수의 수열  $X$ 와 뒤집기 정보가 다음과 같이 주어졌을 때,

- |    |   |    |   |    |   |    |    |    |    |    |
|----|---|----|---|----|---|----|----|----|----|----|
| 위치 | 0 | 1  | 2 | 3  | 4 | 5  | 6  | 7  | 8  | 9  |
| 값  | 3 | 81 | 9 | 12 | 0 | -9 | 36 | 33 | 91 | 10 |
- 입력된 수열  $X$ :
- 입력된 뒤집기 정보(3개): (3, 7) --> (4, 5) --> (0, 4)
- (a, b)는 숫자를 뒤집을 구간을 의미하고, a와 b의 범위는  $0 \sim N-1$ 이고,  $a \leq b$  이다.

1) 뒤집기 정보 (3, 7)에 의해, 수열  $X$ 의 3번째 수부터 7번째 수까지의 순서가 반대로 바뀐다.

위치	0	1	2	3	4	5	6	7	8	9
값	3	81	9	33	36	-9	0	12	91	10

2) 뒤집기 정보 (4, 5)에 의해, 위 수열의 4번째 수부터 5번째 수까지의 순서가 반대로 바뀐다.

위치	0	1	2	3	4	5	6	7	8	9
값	3	81	9	33	-9	36	0	12	91	10

3) 마지막으로 (0, 4)에 의해 순서가 뒤집히고, 최종적으로 만들어지는 수열은 다음과 같다.

위치	0	1	2	3	4	5	6	7	8	9
값	-9	33	9	81	3	36	0	12	91	10

변환된 수열은 한 줄에 출력하되, 줄의 맨 앞에 공백을 하나 출력한다.

입력 예시 1

출력 예시 1

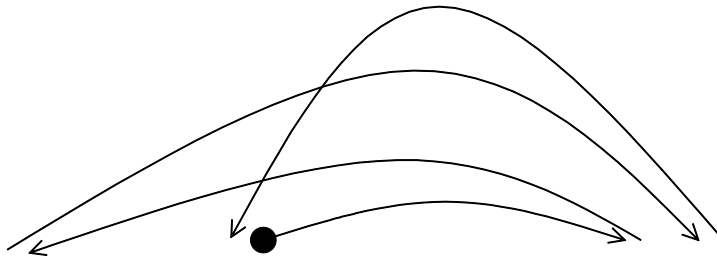
10 ↳ 수열의 길이 N	□ -9 33 9 81 3 36 0 12 91 10 ↳ 최종 수열
3 81 9 12 0 -9 36 33 91 10 ↳ 수열 X	
3 ↳ 뒤집기 정보의 개수	
3 7 4 5 0 4 ↳ 뒤집기 구간 정보	

입력 예시 2

출력 예시 2

6	□ 0 10 20 30 40 50
30 10 20 0 40 50	
2	
1 2 0 3	

**[ 문제 2 ]** (20점)  $N$  ( $2 \leq N \leq 100$ ) 개의 정수로 이루어진 수열  $X$ 를 "위치 바꿈 정보"에 의해 변환한 최종 결과를 출력하는 프로그램을 작성하시오. 위치 바꿈 방식은 다음과 같다. 예를 들어, 10개 정수의 수열  $X$ 와 위치 바꿈 정보가 다음과 같이 주어졌을 때,



○ 수열  $X$ :

위치	0	1	2	3	4	5	6	7	8	9
값	3	81	9	12	0	-9	36	33	91	10

○ 위치 바꿈 정보: 3 --> 8 --> 0 --> 9 --> 3  
(위치 바꿈 정보를 구성하는 수의 범위는 0~N-1이다. 주어지는 위치 바꿈 정보에서 처음과 마지막 위치는 항상 동일하고, 그 외에는 동일한 위치는 없다고 가정하라.)

○ 위 순서 바꿈 정보에 의해, 수열  $X$ 에서  
 3번 위치의 정수 '12'는 8번 위치로 이동,  
 8번 위치의 정수 '91'은 0번 위치로 이동,  
 0번 위치의 정수 '3'은 9번 위치로 이동,  
 9번 위치의 정수 '10'은 3번 위치로 이동 시킨다.

○ 위 변환 규칙에 의해 만들어지는 최종 수열은 다음과 같다.

위치	0	1	2	3	4	5	6	7	8	9
값	91	81	9	10	0	-9	36	33	12	3

변환된 수열은 한 줄에 출력하되, 줄의 맨 앞에 공백을 하나 출력한다.

입력 예시 1

출력 예시 1

10	↪ 입력 수열의 길이 (N)	□ 91 81 9 10 0 -9 36 33 12 3 ↪ 변환 수열
3 81 9 12 0 -9 36 33 91 10	↪ 수열 X	
5	↪ 순서 바꿈 정보의 길이	
3 8 0 9 3	↪ 순서 바꿈 정보	

입력 예시 2

출력 예시 2

6	□ 0 10 20 30 40 50
0 20 40 30 10 50	
4	
1 2 4 1	

**[ 문제 3 ]**  $N \times N$  ( $1 \leq N \leq 100$ ) 크기의 행렬에  $1 \sim N^2$  의 수를 아래 그림과 같이 차례로 위에서 부터  $\rightarrow$  방향과  $\leftarrow$  방향을 번갈아 가면서 채운 결과를 출력하시오.

4 x 4 행렬 :

1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

입력 예시 1

출력 예시 1

4	$\mapsto$ 행렬 크기 N	<input type="checkbox"/> 1 2 3 4 $\mapsto$ 한 줄에 한 행씩 출력 <input type="checkbox"/> 8 7 6 5 <input type="checkbox"/> 9 10 11 12 <input type="checkbox"/> 16 15 14 13
---	-------------------	--

**[ 문제 4 ]**  $N \times M$  ( $1 \leq N, M \leq 100$ ) 크기의 행렬에  $1 \sim MN$  의 수를 아래 그림과 같이 나선형으 로 채운 결과를 출력하시오.

4 x 4 행렬 :

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

4 x 5 행렬 :

1	2	3	4	5
14	15	16	17	6
13	20	19	18	7
12	11	10	9	8

입력 예시 1

출력 예시 1

4 5	$\mapsto$ 행렬 크기 N, M	<input type="checkbox"/> 1 2 3 4 5 $\mapsto$ 한 줄에 한 행씩 출력 <input type="checkbox"/> 14 15 16 17 6 <input type="checkbox"/> 13 20 19 18 7 <input type="checkbox"/> 12 11 10 9 8
-----	----------------------	--

**[ 문제 5 ]**  $N \times M$  ( $1 \leq N, M \leq 100$ ) 크기의 행렬에  $1 \sim MN$  의 수를 아래 그림과 같이  $\swarrow$  대각선 방향으로 채운 결과를 출력하시오.

4 x 4 행렬 :

1	2	4	7
3	5	8	11
6	9	12	14
10	13	15	16

4 x 5 행렬 :

1	2	4	7	11
3	5	8	12	15
6	9	13	16	18
10	14	17	19	20

입력 예시 1

출력 예시 1

4 5	↳ 행렬 크기 N, M	<input type="checkbox"/> 1 2 4 7 11	↳ 한 줄에 한 행씩 출력
		<input type="checkbox"/> 3 5 8 12 15	
		<input type="checkbox"/> 6 9 13 16 18	
		<input type="checkbox"/> 10 14 17 19 20	

문제 1

4 x 4 행렬 :

1			
9	2		
8	10	3	
7	6	5	4

5 x 5 행렬 :

1				
12	2			
11	13	3		
10	15	14	4	
9	8	7	6	5

입력 예시 1

출력 예시 1

4	↦ n만 입력	1 9 2 8 10 3 7 6 5 4
---	---------	-------------------------------

문제2

예시) 행렬 1 : 3 \* 2

행렬 2 : 2 \* 3

1	2
4	5
1	2

X

1	2	3
3	2	1

=

7	6	5
19	18	17
7	6	5

입력 예시 1

출력 예시 1

3 2	↦ 첫째 행렬의 행과 열	7 6 5
2 3	↦ 둘째 행렬의 행과 열	19 18 17
1 2	↦ 첫째 행렬 입력	7 6 5
4 5		
1 2		
1 2 3	↦ 둘째 행렬 입력	
3 2 1		

## 〈자료구조 실습〉 - 연결리스트

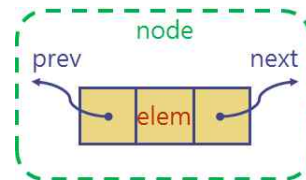
### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서  $\mapsto$  이 후는 각 입력과 출력에 대한 설명이다.

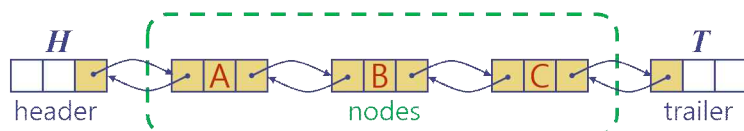
**연결리스트 참고사항 :** 아래 나오는 이중연결리스트/헤더/트레일러 노드 방식을 이용해도 되고, 우리 교재에 나오는 방식을 이용해도 됩니다.

### 1. 연결리스트 구조

- 각 노드에 저장되는 정보
  - elem: 원소
  - prev: 이전 노드를 가리키는 링크
  - next: 다음 노드를 가리키는 링크



- 헤더 및 트레일러 노드
  - 데이터를 가지지 않는 특별 노드



### 2. 이중연결리스트 초기화

- 초기에는 헤더 및 트레일러 노드만 존재
- $O(1)$  시간 소요



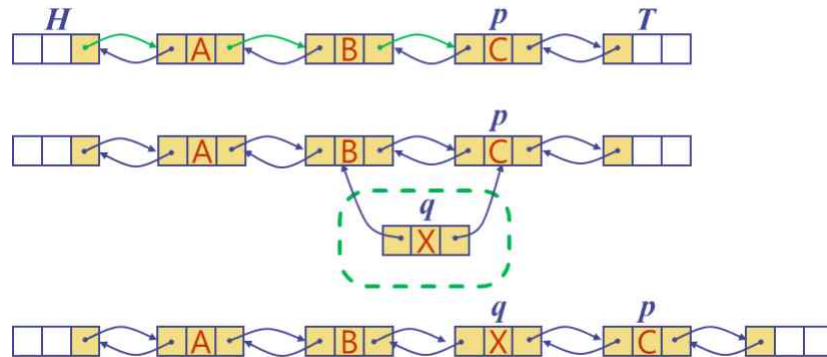
### 3. 이중연결리스트 순회

- 연결리스트의 모든 원소들을 방문
- 순회하면서 필요한 작업 수행(예를 들면 출력)
- $O(n)$  시간 소요



### 4. 이중연결리스트에서 삽입

- 이중연결리스트의 지정된 순위  $r$ 에 원소  $e$ 를 삽입
- $O(n)$  시간 소요



##### 5. 이중연결리스트에서 삭제

- 이중연결리스트로부터 지정된 순위  $r$ 의 노드를 삭제하고, 원소를 반환
- $O(n)$  시간 소요

※ 참고: 초기화, 순회, 삽입, 삭제에 관한 상세 알고리즘은 교재를 참고

[ 문제 1 ] 이중연결리스트를 이용하여 아래의 영문자 연산을 구현하시오.

- 다음 네 가지 연산을 지원해야 함 (순위는 1부터 시작한다고 가정)
  - **add(list, r, e)** : list의 순위  $r$ 에 원소  $e$ 를 추가한다.
  - **delete(list, r)** : list의 순위  $r$ 에 위치한 원소를 삭제한다
  - **get(list, r)** : list의 순위  $r$ 에 위치한 원소를 반환한다.
  - **print(list)** : list의 모든 원소를 저장 순위대로 공백없이 출력한다.

※ 순위 정보가 유효하지 않으면 화면에 에러 메시지 "invalid position" 출력하고, 해당 연산을 무시한다.

- 입력에 대한 설명 (아래 입출력 예시 참조)
  - 각 연산의 내용이 한 줄에 한 개씩 입력되고, 한 개의 줄에는 연산의 종류, 순위, 원소 순서로 입력된다.
  - **연산의 종류**: 연산 이름의 맨 앞 영문자가 대문자 **A, D, G, P**로 주어진다.
  - **순위**: 양의 정수
  - **원소**: 영문자(대문자, 소문자 모두 가능)



입력 예시 1

출력 예시 1

5	↳ 연산의 개수: 5	
A 1 S	↳ add(list, 1, 'S')	
A 2 t	↳ add(list, 2, 't')	
A 3 r	↳ add(list, 3, 'r')	
A 3 a	↳ add(list, 3, 'a')	
P	↳ print(list)	Star      ↳ 연산 p에 의한 출력

입력 예시 2

출력 예시 2

9	↳ 연산의 개수: 9	
A 1 D	↳ add(list, 1, 'D')	
A 2 a	↳ add(list, 2, 'a')	
A 3 y	↳ add(list, 3, 'y')	
D 1	↳ delete(list, 1)	
P	↳ print(list)	ay
G 3	↳ get(list, 3)	invalid position
A 1 S	↳ add(list, 1, 'S')	
P	↳ print(list)	Say
G 3	↳ get(list, 3)	y

**다항식 덧셈 참고사항 :** 연결리스트를 이용하여 구현하되, 아래 설명 방식을 이용해도 되고, 우리 교재에 나오는 방식을 이용해도 됩니다.

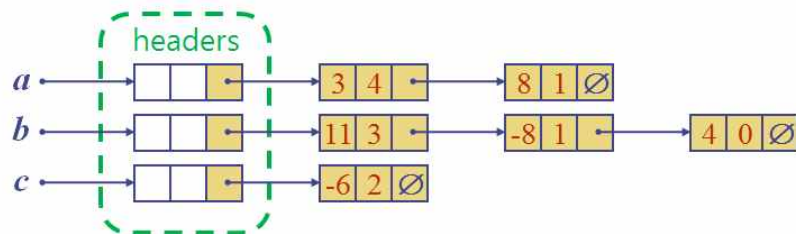
#### 1. 다항식을 표현하는 연결리스트 구조

- 하나의 다항식(polynomial)을 하나의 헤더 단일연결리스트로 표현하는 방식 사용
- 다항식의 각 항은 하나의 노드로 표현하고, 각 노드에는 다음 세 개의 필드를 저장
  - coef: 항의 계수
  - exp: 항의 차수
  - next: 다음 노드를 가리키는 링크
- 하나의 연결리스트의 각 노드는 차수의 내림차순 순으로 유지하고, 계수가 0인 항의 노드는 유지하지 않음
- 예) 아래 세 개의 다항식을 나타내는 단일연결리스트 그림
 
$$a = 3x^4 + 8x$$

$$b = 11x^3 - 8x + 4$$

$$c = -6x^2$$

### polynomials



#### 2. 다항식에 항 추가

- 기존 다항식의 마지막 항을 표현하는 노드 k에 계수 c와 차수 e로 이루어진 새 항 추가

**Alg** appendTerm(k, c, e)

**input** last term of a polynomial expression k, coefficient c, exponent e

**output** cxe appended to k

1.  $t \leftarrow \text{getnode}()$
2.  $t.\text{coef} \leftarrow c$
3.  $t.\text{exp} \leftarrow e$
4.  $t.\text{next} \leftarrow \text{NULL}$
5.  $k.\text{next} \leftarrow t$
6.  $k \leftarrow t$  {k advances to t}
7. return

#### 3. 다항식 덧셈

- 두 개의 다항식 x, y에 대한 덧셈을 수행하여 그 결과를 새로운 헤더 단일연결리스트에 저장
  - 예: 위 예의 다항식 a, b의 덧셈 결과는  $3x^4 + 11x^3 + 4$  를 반환

```

Alg addPoly(x, y)
  input polynomial expression x, y
  output x + y
1. result ← getnode()      {new header }
2. result.next←NULL        {may be null }
3. i ← x.next
4. j ← y.next
5. k ← result
6. while ((i ≠ NULL) & (j ≠ NULL))
    if (i.exp > j.exp)
        appendTerm(k, i.coef, i.exp)
        i ← i.next
    else if (i.exp < j.exp)
        appendTerm(k, j.coef, j.exp)
        j ← j.next
    else
        sum ← i.coef + j.coef
        if (sum ≠ 0)
            appendTerm(k, sum, i.exp)
            i ← i.next
            j ← j.next
7. while (i ≠ NULL )
    appendTerm(k, i.coef, i.exp)
    i ← i.next
8. while (j ≠ NULL)
    appendTerm(k, j.coef, j.exp)
    j ← j.next
9. return result

```

**[ 문제 2 ]** 다항식의 덧셈을 구하는 프로그램을 작성하라.

- 입력에 대한 설명 (아래 입출력 예시 참조)
  - 첫 번째 다항식의 항의 개수가 입력되고, 이후에 다항식의 각 항의 (계수, 지수) 쌍이 지수의 내림차순으로 입력됨
  - 동일한 방식으로 두 번째 다항식의 정보가 입력됨
- 출력에 대한 설명 (아래 입출력 예시 참조)
  - 결과 다항식의 각 항의 (계수, 지수) 쌍을 지수의 내림차순으로 출력

입력 예시 1

출력 예시 1

3	↪ 첫 번째 다항식의 항의 개수	□ 2 6 7 3 3 2 3 1 1 0	↪ $2x^6 + 7x^3 + 3x^2 + 3x + 1$
5 3 3 2 3 1	↪ $5x^3 + 3x^2 + 3x$		
3	↪ 두 번째 다항식의 항의 개수		
2 6 2 3 1 0	↪ $2x^6 + 2x^3 + 1$		

입력 예시 2

출력 예시 2

2	↪ 첫 번째 다항식의 항의 개수	□ -3 10 5 7	↪ $-3x^{10} + 5x^7$
2 7 3 0	↪ $2x^7 + 3$		
3	↪ 두 번째 다항식의 항의 개수		
-3 10 3 7 -3 0	↪ $-3x^{10} + 3x^7 - 3$		

## 〈자료구조 실습〉 - 집합

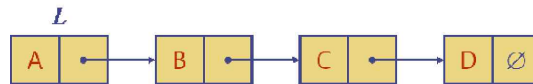
### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서  $\mapsto$  이 후는 각 입력과 출력에 대한 설명이다.

**[ 문제 1 ]** 두 개의 집합 A와 B를 입력 받아, A가 B의 부분집합인지를 검사하는 프로그램을 작성하시오.

### 주의:

- 1) 집합은 오름차순 양의 정수로 저장 및 출력되어야 한다.
- 2) 공집합은 공집합을 포함한 모든 집합의 부분집합이다.
- 3) **입력:** 프로그램은 두 개의 집합 A, B를 차례로 표준입력 받는다. 한 개의 집합을 나타내는 두 개의 입력 라인은 다음과 같이 구성된다.  
 첫 번째 라인: 정수 n (집합 크기, 즉 집합 원소의 개수)  
 두 번째 라인: 집합의 원소들 (오름차순 양의 정수 수열).  
 공집합은 첫 번째 라인은 0, 두 번째 라인은 존재하지 않는다.
- 4) **출력:**  $A \subset B$ 이면 0을 출력하고, 그렇지 않으면 집합 B에 속하지 않은 집합 A의 가장 작은 원소를 표준 출력한다.
- 5) 모든 집합은 헤더 노드가 없는 단일연결리스트(singly-linked list) 형태로 구축되어야 한다.
- 6) **참고:** 아래 그림은 일반적인 단일연결리스트를 나타낸다. 빈 리스트의 경우 null pointer로 나타낸다. (그림의 노드에 저장된 원소가 영문자인데, 이는 무시하고 리스트의 형태만 참고하시오.)



입력 예시 1

3	$\mapsto$ 집합 A 크기	0	$\mapsto A \subset B$
4 6 13	$\mapsto$ 집합 A		
6	$\mapsto$ 집합 B 크기		
1 3 4 6 8 13	$\mapsto$ 집합 B		

출력 예시 1

입력 예시 2

3	$\mapsto$ 집합 A 크기	53	$\mapsto A \not\subset B$
7 10 53	$\mapsto$ 집합 A		
4	$\mapsto$ 집합 B 크기		
7 10 15 45	$\mapsto$ 집합 B		

출력 예시 2

입력 예시 3

0	↪ 집합 A (공집합)
3	↪ 집합 B 크기
9 20 77	↪ 집합 B

출력 예시 3

0	↪ $A \subset B$
---	-----------------

입력 예시 4

0	↪ 집합 A (공집합)
0	↪ 집합 B (공집합)

출력 예시 4

0	↪ $A \subset B$
---	-----------------

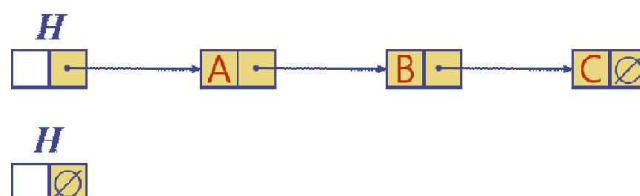
다음 함수를 작성하여 사용하시오.

- 함수 subset : 집합 A가 집합 B의 부분집합인지 여부 검사
  - 인자: 양의 정수 집합 A, B (A, B는 각각 단일연결리스트의 헤드 노드)
  - 반환값: 정수 (A  $\subset$  B면 0, 그렇지 않으면 집합 B에 속하지 않은 집합 A의 가장 작은 원소)

**[ 문제 2 ]** 두 개의 집합을 입력받아, 합집합과 교집합을 구하는 프로그램을 작성하시오.

**주의:**

- 1) 모든 집합은 오름차순 양의 정수로 저장 및 출력되어야 한다.
- 2) 공집합 처리에 주의
- 3) **입력:** 프로그램은 두 개의 집합 A, B를 차례로 표준입력 받는다. 한 개의 집합을 나타내는 두 개의 입력 라인은 다음과 같이 구성된다.
  - 첫 번째 라인: 정수 n (집합 크기, 즉 집합 원소의 개수)
  - 두 번째 라인: 집합의 원소들 (오름차순 양의 정수 수열).
 따라서 공집합은 첫 번째 라인은 0, 두 번째 라인은 존재하지 않는다.
- 4) **출력:** 각 연산 결과는 두 개의 라인으로 표준출력한다. 첫 번째 라인은 합집합을, 두 번째 라인은 교집합을 나타낸다. 이때 공집합은 0로 출력한다.
- 5) 모든 집합은 헤더(header) 노드가 추가된 단일연결리스트 형태로 구축되어야 한다.
- 6) **참고:** 아래 첫 번째 그림은 일반적인 헤더 단일연결리스트를, 아래 두 번째 그림은 빈 리스트를 나타낸다. (그림의 노드에 저장된 원소가 영문자인데, 이는 무시하고 리스트의 형태만 참고하시오.)



입력 예시 1

6	↳ 집합 A 크기
3 7 45 88 99 101	↳ 집합 A
4	↳ 집합 B 크기
7 10 15 45	↳ 집합 B

출력 예시 1

<input type="checkbox"/> 3 7 10 15 45 88 99 101	↳ 합집합
<input type="checkbox"/> 7 45	↳ 교집합

입력 예시 2

0	↳ 집합 A 크기 (공집합)
3	↳ 집합 B 크기
9 20 77	↳ 집합 B

출력 예시 2

<input type="checkbox"/> 9 20 77	↳ 합집합
<input type="checkbox"/> 0	↳ 교집합 (공집합)

입력 예시 3

0	↳ 집합 A 크기 (공집합)
0	↳ 집합 B 크기 (공집합)

출력 예시 3

<input type="checkbox"/> 0	↳ 합집합 (공집합)
<input type="checkbox"/> 0	↳ 교집합 (공집합)

다음 함수를 작성하여 사용하시오.

- 함수 union: 합집합 연산
  - 인자: 양의 정수 집합 A, B (A, B는 각각 헤더 단일연결리스트의 헤더 노드)
  - 반환값:  $A \cup B$ 의 헤더 노드 주소 또는, 공집합인 경우 빈 리스트(즉, 헤더 노드만 존재)
- 함수 intersect: 교집합 연산
  - 인자: 양의 정수 집합 A, B (A, B는 각각 헤더 단일연결리스트의 헤더 노드)
  - 반환값:  $A \cap B$ 의 헤더 노드 주소 또는, 공집합인 경우 빈 리스트(즉, 헤더 노드만 존재)

## 〈자료구조 실습〉 - 스택 (1)

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↪ 이 후는 각 입력과 출력에 대한 설명이다.

### [ 문제 1 ] 다음의 스택 ADT를 배열로 구현하고 테스트하는 프로그램을 작성하세요

- 데이터: 영문자
- 다음의 연산을 지원해야 함
  - push(stack, 'c') : stack의 top에 데이터를 추가한다. stack이 이미 꽉 차있으면 해당 데이터는 스택에 저장하지 않고 "Stack FULL"을 출력한다.
  - pop (stack) : stack의 top에 있는 데이터를 반환하고 stack에서 제거한다. stack이 비어 있으면 "Stack Empty"를 출력한다.
  - peek(stack): stack의 top에 있는 데이터를 화면에 출력한다. stack은 변화하지 않는다. stack이 비어 있으면 "Stack Empty"를 출력한다.
  - duplicate(stack): stack의 top에 있는 데이터를 pop해서 두 번 push 한다. stack이 이미 꽉 차있으면 "Stack FULL"을 출력한다.
  - upRotate(stack, n): stack의 맨 위 n 개의 데이터를 회전시킨다. 예를 들면 n 이 3이고 stack의 top에서부터 elem1, elem2, elem3, .... 이 저장되어 있으면 데이터를 하나씩 위쪽으로 이동시킨다. 맨 위쪽 (top)의 elem1은 n-1번 아래쪽으로 이동해서 스택의 결과는 elem2, elem3, elem1, ...이된다.  
단, n이 데이터의 개수보다 큰 경우에는 아무 작업을 하지 않는다.
  - downRotate(stack, n): stack의 맨 위 n 개의 데이터를 회전시킨다. 예를 들면 n 이 3이고 stack의 top에서부터 elem1, elem2, elem3, .... 이 저장되어 있으면 데이터를 하나씩 d 아래쪽으로 이동시킨다. top에서부터 n번째의 데이터는 top으로 이동해서, 스택의 결과는 elem3, elem1, elem2, ...이된다.  
단, n이 데이터의 개수보다 큰 경우에는 아무 작업을 하지 않는다.
  - print(stack) : stack의 모든 데이터를 top에서부터 순서대로 공간없이 출력한다.
- 입력에 대한 설명 (아래 입출력 예시 참조)
  - 각 연산의 내용이 한 줄에 하나씩 입력되고, 하나의 줄에는 연산의 종류와 그에 필요한 데이터가 입력된다.
  - 연산의 종류: 각 연산 이름은 POP, PUSH , PEEK, DUP, UpR, DownR, PRINT로 주어진다.

입력 예시 1

4	↳ stack의 크기 N
10	↳ 연산의 개수
POP	↳ pop(stack)
PUSH s	↳ push(stack, 's')
PUSH t	↳ push(stack, 't')
PUSH a	↳ push(stack, 'a')
PUSH r	↳ push(stack, 'r')
PRINT	↳ print(stack)
UpR 3	↳ upRotate(stack, 3)
PRINT	↳ print(stack)
PUSH s	↳ push(stack, 's')
PEEK	↳ push(stack)

출력 예시 1

Stack Empty	↳ 1번 POP 연산의 결과
rats	↳ 6번 PRINT 연산의 결과
atrs	↳ 8번 PRINT 연산의 결과
Stack FULL	↳ 9번 PUSH 연산의 결과
a	↳ 10번 PEEK 연산의 결과

입력 예시 2

5	↳ stack의 크기 N
11	↳ 연산의 개수
PUSH s	↳ push(stack, 's')
PUSH r	↳ push(stack, 'r')
PUSH a	↳ push(stack, 'a')
PUSH t	↳ push(stack, 't')
PUSH s	↳ push(stack, 's')
PRINT	↳ print(stack)
DownR 4	↳ downRotate(stack, 4)
PRINT	↳ print(stack)
POP	↳ pop(stack)
POP	↳ pop(stack)
PRINT	↳ print(stack)

출력 예시 2

stars	↳ 6번 PRINT 연산의 결과
rstas	↳ 8번 PRINT 연산의 결과
tas	↳ 11번 PRINT 연산의 결과

입력 예시 3

3	↳ stack의 크기 N
5	↳ 연산의 개수
PUSH d	↳ push(stack, 'd')
DUP	↳ duplicate(stack)
PUSH a	↳ push(stack, 'a')
PRINT	↳ print(stack)
PUSH s	↳ push(stack, 's')

출력 예시 3

add	↳ 4번 PRINT 연산의 결과
Stack FULL	↳ 5번 PRINT 연산의 결과



[ 문제 2 ] 스택의 응용으로 키보드로부터 입력된 한 줄의 수식문장에서 괄호 짝의 유효성을 검사하는 프로그램을 작성하세요. 괄호짝은 { }, [ ], ( ) 의 3 종류를 갖는다.

주의: 수식문장은 1000개의 문자를 넘지 않는다. 수식문장은 공백문자를 포함할 수 있다.

출력은 유효하지 않으면 'Wrong\_N' 유효하면 'OK\_N'를 출력한다. 여기서 N은 문장안의 괄호의 개수이다.

입력 예시 1

(3+40\*(2+(30-7)\*2133)

출력 예시 1

Wrong\_5

입력 예시 2

3\*{4+(2-792)/1} + [3\*{4-2\* (100 -7)}]

출력 예시 2

OK\_10

입력 예시 3

301\*{4+(2101-7)/1} + 9\*{4-2\* (10108-7)}]}

출력 예시 3

Wrong\_9

입력 예시 4

(3\*{4001+(2-7)/1} + [3\*{4-2\* (1-7)}])

출력 예시 4

OK\_12

## 〈자료구조실습〉 - 스택 (2)

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

### [ 문제 1 ] 스택을 이용하여 중위수식을 후위수식으로 변환하는 프로그램을 작성하시오

- 스택은 배열이나 연결리스트로 구현함
- 수식의 피연산자는 영문자(대문자)로 나타내고, 각 수식의 최대길이는 100으로 함
- 수식은 아래 우선순위를 갖는 연산자들을 포함함 (숫자가 높을수록 우선순위가 높음)

입력토큰	연산자	우선순위
! + -	단항연산자	6
*	곱셈	5
/	나눗셈	5
+	덧셈	4
-	뺄셈	4
>	관계연산자	3
<	관계연산자	3
&&	논리연산자(AND)	2
	논리연산자(OR)	1

- 같은 우선순위를 갖는 연산자들은 왼쪽에서 오른쪽으로 계산하도록 함
- 입출력에 대한 설명 (아래 입출력 예시 참조)

1) 첫 번째 라인 : 수식의 개수

2) 두 번째 라인 :

- 하나의 줄에 수식이 공백 없이 입력됨

입력 예시 1

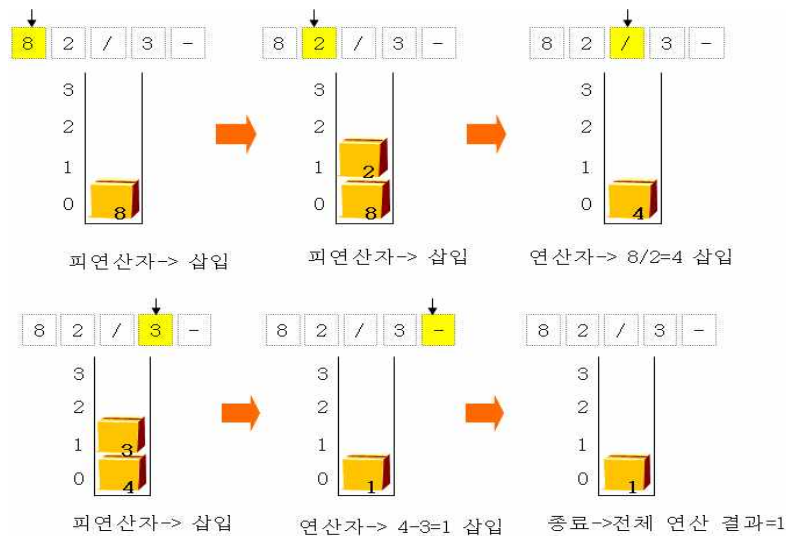
출력 예시 1

5	↳ 수식의 개수	AB*C+DE+F*+
A*B+C+(D+E)*F	↳ 첫 번째 수식	ABC*+
A+B*C	↳ 두 번째 수식	AB/C-DE*+FG*-
A/B-C+D*E-F*G	↳ 세 번째 수식	ABC*D+E*+
A+(B*C+D)*E	↳ 네 번째 수식	AB&&C  EF>
A&&B  C  !(E>F)	↳ 다섯 번째 수식	

[ 문제 2 ] 후위로 변환된 수식을 입력받아 스택을 사용하여 계산한 후 결과 값을 출력하는 프로그램을 작성하시오

- 스택은 배열이나 연결리스트로 구현함
- 수식의 피연산자는 0에서 9사이의 정수이고, 각 수식의 최대길이는 100으로 함
- 수식의 연산자는 곱하기, 나누기, 더하기, 빼기로 구성되며, 정수 연산 수행
  - 즉, 나누기의 경우, 몫 계산

※ 예제 : 82/3-



- 입출력에 대한 설명 (아래 입출력 예시 참조)

- 1) 첫 번째 라인 : 수식의 개수
- 2) 두 번째 라인 :
  - 하나의 줄에 후위수식이 공백 없이 입력됨

입력 예시 1

4	↳ 수식의 개수	35	↳ 5*3+2+(6+3)*2의 결과
53*2+63+2*+	↳ 첫 번째 수식	14	↳ 2+3*4의 결과
234*+	↳ 두 번째 수식	6	↳ 7/2-3+4*2-2*1의 결과
72/3-42*+21*-	↳ 세 번째 수식	23	↳ 9+(2*3+1)*2의 결과
923*1+2*+	↳ 네 번째 수식		

출력 예시 1

## 〈자료구조 실습〉 - 큐

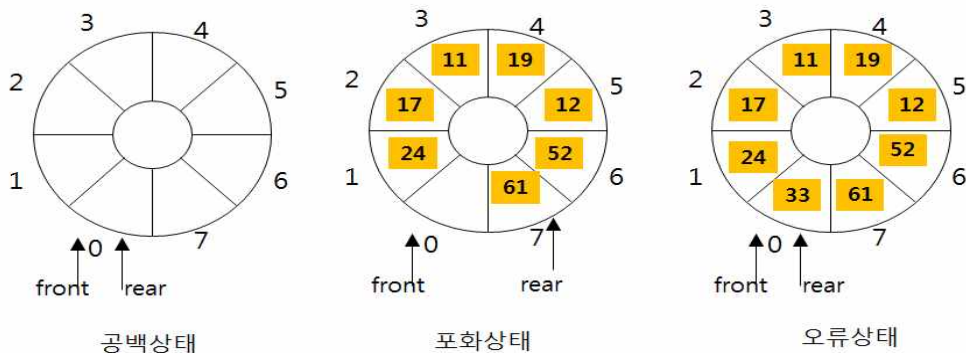
### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

**[ 문제 1-큐 ]** 배열로 구성된 원형 큐에서의 삽입, 삭제 프로그램을 작성하고, 입력된 순서대로 삽입과 삭제가 이루어 졌을 경우, 원형 큐에 있는 원소 값을 차례대로 출력하시오.

- 문제의 원형 큐에서는 **포화 상태와 공백 상태를 구분**하기 위해 한 자리를 비워둠
  - front, rear, 배열의 초기 값은 0
  - rear의 값을 하나 증가시킨 후 데이터를 큐에 삽입 (출력 예시 1 참고). 또한, front의 값을 하나 증가시킨 후 front가 가리키는 데이터를 삭제함
  - front == rear이면 공백상태로 정의하고, front가 rear보다 하나 앞에 있으면 포화 상태로 정의 함

※ 단, 위 초기 값 및 조건은 하나의 구현 예시이고(교재마다 다를 수 있음), 다른 방법 정의를 사용해도 무방하나, **초기 상태에서 맨 처음 삽입되는 위치는 0번이 아니고, 1번이 되어야 함** (그렇지 않으면 문제의 입출력 예시와 다른 결과가 나올 수 있음)



### ○ 입출력 형식:

- 1) 첫 번째 라인 : 양의 정수 q (원형 큐의 크기)
  - ※ q 값에는 제한이 없다. 또한, 동적 할당을 사용하여야 함
- 2) 두 번째 라인 : 양의 정수 n (연산의 개수)
- 3) 세 번째 이후 라인: n개의 연산이 차례로 입력됨.
  - ※ 연산의 종류는 I (삽입), D (삭제), P (출력)
  - I 10 : 원형 큐에 원소 10을 삽입 함 (큐 원소는 양의 정수)
  - D : 원형 큐에서 데이터를 삭제 함. 또한, 해당 배열의 값을 0으로 함
  - P : **배열에 있는 모든 값들을** 차례로 화면에 출력 함 (입출력 예시 참조)

- ※ **overflow 발생 시** (즉, 포화상태에서 삽입연산이 발생하는 경우),  
화면에 overflow 와 배열 값들을 모두 출력하고 프로그램 종료
- ※ **underflow 발생 시** (즉, 공백상태에서 삭제 연산이 발생하는 경우),  
화면에 underflow 를 출력하고 프로그램 종료

입력 예시 1

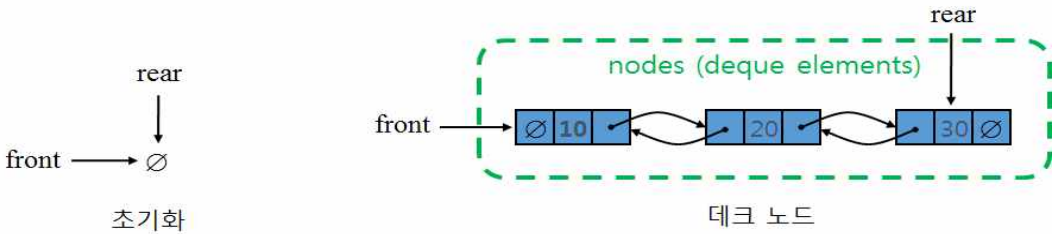
6	↳ q = 6
10	↳ n = 10
I 10	↳ 삽입
I 20	↳ 삽입
P	↳ 화면출력
I 30	↳ 삽입
I 40	↳ 삽입
D	↳ 삭제
P	↳ 화면출력
I 50	↳ 삽입
I 60	↳ 삽입
I 70	↳ 삽입

출력 예시 1

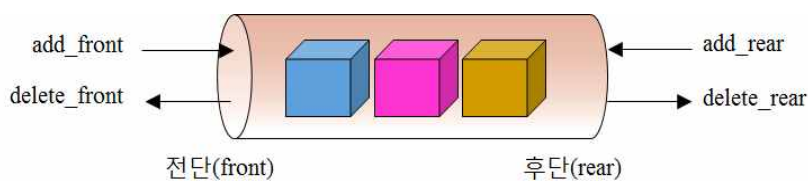
0 10 20 0 0 0	↳ 3번째 연산(P)에 의한 출력
0 0 20 30 40 0	↳ 7번째 연산(P)에 의한 출력
overflow 60 0 20 30 40 50	↳ 10번째 연산(I 70)에서 overflow 발생

[ 문제 2-데크 ] 데크는 큐의 전단 (front)와 후단 (rear)에서 모두 삽입과 삭제가 가능한 자료구조이다. 이중연결리스트를 사용하여 아래에 정의된 데크 함수들을 구현하시오.

- 본 문제에서는 **헤드 노드와 테일 노드가 없는 이중연결리스트를 사용할 것**
- 주의!! 연산 수행 도중에 원소가 모두 삭제되어 데크에 원소가 없는 경우에도, 아래 초기화 상태가 되어야 함



- 데크 연산
- add\_front(deque, X) : deque의 앞에 데이터 X를 추가함
- add\_rear(deque, X) : deque의 뒤에 데이터 X를 추가함
- delete\_front(deque) : deque의 앞에 있는 데이터를 반환한 다음 삭제함
- delete\_rear(deque) : deque의 뒤에 있는 데이터를 반환한 다음 삭제함
- print(deque) : deque의 모든 데이터들을 전단부 부터 저장된 순서대로 출력함



- **입출력 형식:**
  - 1) 첫 번째 라인 : 연산의 개수 n
  - 2) 두 번째 이후 라인: n개의 연산이 한 줄에 하나씩 차례로 입력됨
    - 하나의 줄에는 연산의 종류, 추가인 경우 원소가 주어짐 (원소는 양의 정수로 표기)
    - 연산의 종류: 다음의 연산 이름이 대문자로 주어짐
      - AF (add\_front), AR (add\_rear), DF (delete\_front), DR (delete\_rear), P (print)
- ※ underflow 발생 시, 화면에 underflow 를 출력하고 프로그램 종료

입력 예시 1	출력 예시 1
7            ↪ 연산의 개수	<input type="checkbox"/> 20 10 30        ↪ 4번째 연산(P)에 의한 출력
AF 10        ↪ add_front(deque, 10)	<input type="checkbox"/> 10                ↪ 7번째 연산(P)에 의한 출력
AF 20        ↪ add_front(deque, 20)	
AR 30        ↪ add_rear(deque, 30)	
P            ↪ print(deque)	
DF            ↪ delete_front(deque)	
DR            ↪ delete_rear(deque)	
P            ↪ print(deque)	

입력 예시 2	출력 예시 2
15            ↪ 연산의 개수	<input type="checkbox"/> 30 20 10 40 50    ↪ 6번째 연산(P)에 의한 출력
AF 10        ↪ add_front(deque, 10)	<input type="checkbox"/> 10 40              ↪ 10번째 연산(P)에 의한 출력
AF 20        ↪ add_front(deque, 20)	underflow            ↪ 13번째 연산(DR)에서
AF 30        ↪ add_front(deque, 30)	underflow발생. 실행을 종료함
AR 40        ↪ add_rear(deque, 40)	
AR 50        ↪ add_rear(deque, 50)	
P            ↪ print(deque)	
DF            ↪ delete_front(deque)	
DF            ↪ delete_front(deque)	
DR            ↪ delete_rear(deque)	
P            ↪ print(deque)	
DF            ↪ delete_front(deque)	
DR            ↪ delete_rear(deque)	
DR            ↪ delete_rear(deque)	

## 〈자료구조 실습〉 - 트리 (1)

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

### 트리 1주차: 이진 트리 삽입과 탐색

#### [연결리스트를 이용한 이진 트리]

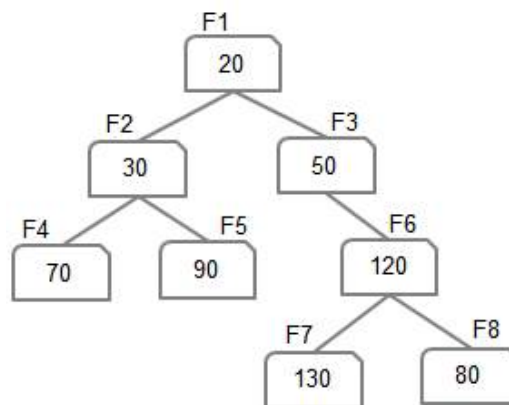
- 이진트리의 노드에 저장되는 정보

- data: 노드에 저장되는 값 (아래 문제에서 폴더의 용량)
- left: 좌측 child 노드를 가리키는 링크
- right: 우측 child 노드를 가리키는 링크

left	data	right

- 이진 트리를 이용한 폴더 구조 표현

- 이진트리는 최대 2개의 자식 노드를 갖음.
- 컴퓨터의 폴더 구조가 이진 트리 형태로 구성되어 있다고 가정함.
- 각각의 노드는 폴더 이름과 용량을 나타내며, 아래 트리에서 폴더 F1에는 20M 가 저장되어 있음을 의미함.

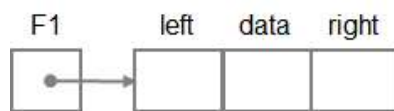


**[ 문제 1 ]** 위 트리를 연결리스트를 이용해서 구현하고, 주어진 노드에 대해 자신과 왼쪽 자식, 우측 자식의 용량을 순서대로 출력하시오.

※ 참고사항: 실습 및 테스트 용이성을 위해 본 문제에서는 고정된 트리를 사용하지만, 일반적으로 삽입, 삭제 가능한 트리를 사용함

**도움말:**

- 루트노드 삽입 함수를 만들어 사용하며, data(폴더 용량), left (왼쪽 자식 링크), right (오른쪽 자식 링크) 를 인수로 받음.
- 모든 노드는 아래 그림과 같이 자신의 위치를 가리키는 포인터변수를 만들어 사용함.



- 단말 노드부터 생성하고, 부모노드를 붙여가는 방식으로 트리를 구성함.
  - 예를 들어, F7과 F8을 생성하고, 이를 이용해 F6 생성하여 F6, F7, F8로 구성된 트리 생성
  - 비슷한 방법으로 트리를 확장해 나감

**출력:**

- 자식 및 노드 존재 여부에 따라 출력 내용이 달라짐
  - 한쪽 자식만 존재하는 경우, 자신과 해당 자식 노드의 용량 2개 값만 출력
  - 자식 노드가 없는 경우, 자신의 용량 1개 값만 출력
  - 존재하지 않는 노드번호가 입력되는 경우 -1을 출력

입력 예시 1

출력 예시 1

2      ↳ 노드번호 (F2을 의미)	30 70 90   ↳ 자신, 왼쪽, 오른쪽 순으로 출력
------------------------	---------------------------------

입력 예시 2

출력 예시 2

3      ↳ 노드번호 (F3을 의미)	50 120   ↳ 왼쪽 자식은 존재하지 않음
------------------------	---------------------------

입력 예시 3

출력 예시 3

4      ↳ 노드번호 (F4을 의미)	70      ↳ 자신의 용량만 출력
------------------------	----------------------

입력 예시 4

출력 예시 4

9      ↳ 노드번호	-1      ↳ F9는 존재하지 않는 노드임.
---------------	----------------------------

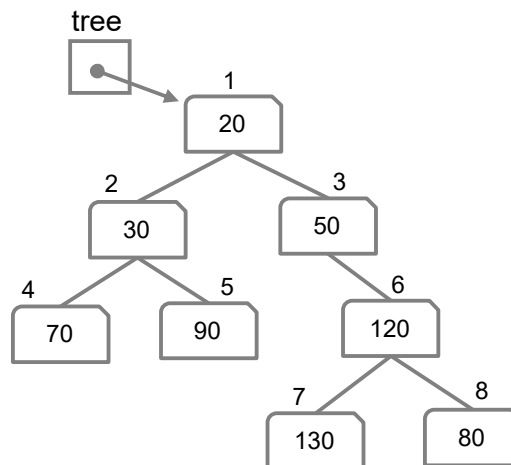


## 〈자료구조 실습〉 - 트리 (2)

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

### 트리 2주차: 트리의 순회



[ 문제 1 ] 위 트리에 대해 순회 방법과 폴더 id가 주어지면, 아래의 트리의 루트노드에서 출발하여 해당 노드를 탐색하여 찾고, 이 노드를 시작점으로 순회하며 각 폴더의 용량을 출력하는 프로그램을 작성하시오.

- 노드 id를 저장하기 위해 노드는 다음과 같은 구조체를 만들어 사용함.
- 지난주 문제의 F1, F2와 같은 노드별 포인터는 사용할 수 없으며, 주어진 노드를 탐색하여 찾아야 함.

left	id	data	right

### 입출력 상세:

- 순회 방법 종류 (입력)
  - 1: 전위순회, 2: 중위순회, 3: 후위순회
- 존재하지 않는 폴더 이름이 입력되는 경우 -1을 **출력**

입력 예시 1

출력 예시 1

1 2 ↳ 1: 전위순회, 노드 id	□30 70 90 ↳ F2에서 전위순회 결과
----------------------	--------------------------

입력 예시 2

출력 예시 2

2 3    ↪ 2: 중위순회, 노드 id	□50 130 120 80    ↪ F3에서 중위순회 결과
-------------------------	----------------------------------

입력 예시 3

출력 예시 3

1 9    ↪ 1: 전위순회, 노드 id	-1                    ↪ F9는 존재하지 않는 노드임.
-------------------------	--

**[ 문제 2 ]** 위 트리에 대해 폴더 id가 주어지면, 해당 폴더의 서브트리의 용량의 합을 계산하는 프로그램을 작성하시오.

- 트리 순회를 이용하여 구현
- 합을 계산할 때 입력된 노드의 용량도 포함
- 존재하지 않는 폴더 이름이 입력되는 경우 -1을 **출력**

입력 예시 1

출력 예시 1

3            ↪ 노드 id	380            ↪ 50+120+130+80
----------------------	--------------------------------

입력 예시 2

출력 예시 2

4            ↪ 노드 id	70            ↪ 70 (F4)
----------------------	-------------------------

입력 예시 3

출력 예시 3

9            ↪ 노드 id	-1
----------------------	----

## 〈자료구조 실습〉 - 트리 (3)

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↳ 이 후는 각 입력과 출력에 대한 설명이다.

### 트리 3주차: 이진 트리 만들기 및 탐색

트리 1, 2주차 실습에서는 트리가 하나로 고정되었는데, 이번 실습에서는 트리가 고정되지 않고 트리의 모양이 입력으로 주어진다.

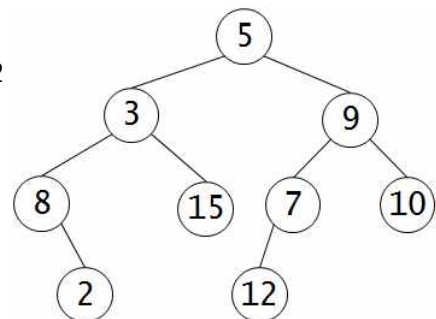
#### 1. 트리 만들기 (구현)

- 트리는 연결이진트리로 구현 (또는 링크 표현법 사용)하고, 각 노드에 저장되는 정보는 아래와 같다.

왼쪽 자식 링크	노드 번호	오른쪽 자식 링크
----------	-------	-----------

- 전위(선위) 순회 순서로 각 노드에 대한 정보가 주어지면, 트리를 루트부터 확장해 가는 방식으로 트리를 구성할 수 있다.
- 노드 번호는 양의 정수로 모두 다르고, 노드 번호에 특별한 순서는 없다.
- 각 노드의 정보는 3개의 정수, (x, y, z)로 표현되는 데, x는 해당 노드의 번호, y는 x의 왼쪽 자식 노드의 번호, z는 x의 오른쪽 자식 노드의 번호를 나타낸다.  
해당되는 자식이 없는 경우에는 0 이 주어진다.

예) 5 3 9    ↳ 5의 왼쪽 자식은 3, 오른쪽 자식은 9  
      3 8 15   ↳ 3의 왼쪽 자식은 8, 오른쪽 자식은 15  
      8 0 2    ↳ 8의 왼쪽 자식은 없고, 오른쪽 자식은 2  
      2 0 0    ↳ (이하 생략)  
      15 0 0  
      9 7 10  
      7 12 0  
      12 0 0  
      10 0 0

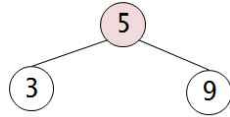


위 노드 정보에서, x에 해당하는 노드 번호를 차례로 쓰면, 전위(선위) 순회 결과가 된다.

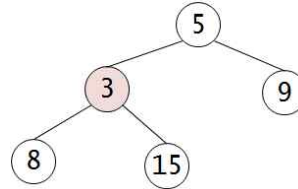
5 3 8 2 15 9 7 12 10

○ 위 예에서 트리가 만들어 지는 과정

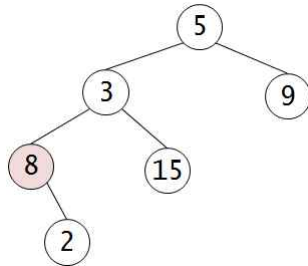
1) 첫 번째 노드 정보 (5 3 9)를  
처리한 후의 트리 모양



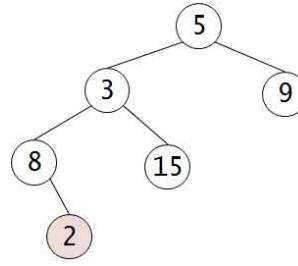
2) 두 번째 노드 정보 (3 8 15)까지  
처리한 후의 트리 모양



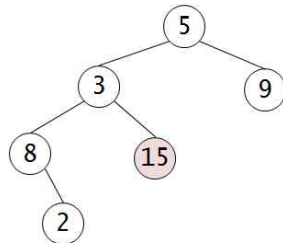
3) 세 번째 노드 정보 (8 0 2)까지  
처리한 후의 트리 모양



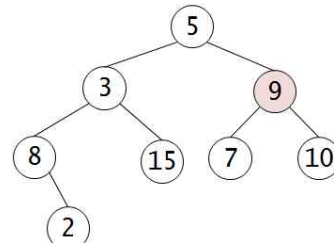
4) 네 번째 노드 정보 (2 0 0)까지  
처리한 후의 트리 모양



5) 다섯 번째 노드 정보 (15 0 0)까지  
처리한 후의 트리 모양



6) 다섯 번째 노드 정보 (9 7 10)까지  
처리한 후의 트리 모양



(이 후 과정 생략)

## 2. 트리 탐색

○ 트리 탐색은 루트(root) 노드에서 시작하여, 자식 링크를 따라 내려가면서 진행됨

- 탐색 도중 만나는 노드에서 어느 자식을 따라 내려가는 지 정보가 주어지면,  
탐색 중 방문하는 노드의 번호들은 유일하게 결정됨.

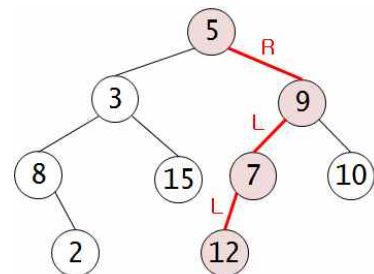
예) 탐색 정보가 아래와 같이 주어지면 (L은 왼쪽 자식, R은 오른쪽 자식을 의미),

RLL

탐색 중 방문하는 노드의 번호를 순서대로 적으면,

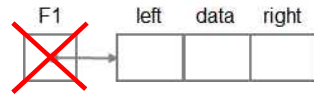
5 9 7 12

가 된다. (오른쪽 그림 참조)



**[ 문제 1 ]** 위에서 설명한 방식대로 트리 정보와 탐색 정보가 주어졌을 때, 트리를 생성하고 탐색 도중 방문하는 노드의 번호를 차례로 출력하는 프로그램을 작성하시오.

- 트리 1주차 실습에서처럼 모든 노드마다 자신의 위치를 가리키는 포인터변수를 만들어 사용하면 안 됨.



- 오직 루트(root) 노드에 대해서만 허용. 즉, 트리는 루트 노드를 통해서만 접근 가능

#### 입력 상세:

- 트리 정보
  - 첫 째 줄에 노드의 개수  $n$ 이 주어진다.
  - 다음  $n$ 개의 줄에, 전위(선위) 순회 순서로 노드의 정보가 주어진다. (위 설명 참조)
- 탐색 정보 (트리 정보가 모두 주어진 후)
  - 탐색 횟수  $s$ 가 주어진다.
  - 다음  $s$ 개의 줄에, 탐색 정보가 주어진다. (각 탐색은 루트 노드에서 새로 시작)
  - 하나의 탐색 정보는 공백없이, 'L'과 'R'로 구성된 문자열(**최대 길이 100**)로 주어진다.
  - 유효하지 않은 탐색 정보는 주어지지 않는다. 예를 들어, 위 트리에서 "RRR" 과 같은 탐색 정보는 유효하지 않다. 두 번 오른쪽 자식을 따라 내려가면 노드 10인데, 노드 10의 오른쪽 자식은 정의되지 않았다.

#### 출력 상세:

- 탐색 시 방문하는 노드의 번호를 순서대로 출력한다. (하나의 줄에 한 번의 탐색 결과 출력)

##### 입력 예시 1

```

9           ↳ 노드 개수
5 3 9
3 8 15
8 0 2
2 0 0
15 0 0
9 7 10
7 12 0
12 0 0
10 0 0
3           ↳ 탐색 횟수
RLL
LL
LR
    
```

##### 출력 예시 1

```

□5 9 7 12   ↳ 첫 번째 탐색 결과
□5 3 8       ↳ 두 번째 탐색 결과
□5 3 15      ↳ 두 번째 탐색 결과
    
```