Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

시분초 입력받아 분만 출력하기

입력되는 시:분:초 에서 분만 출력해보자.

참고
```
int h, m, s;
scanf("%d:%d:%d", &h, &m, &s);
```

를 실행하면 콜론을 사이에 둔 형식으로 입력되어, h, m, s에 각각 정수값만 저장된다.

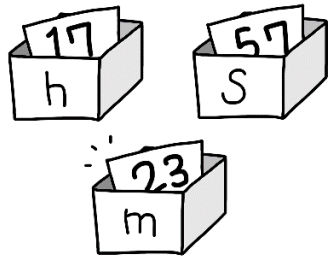
◎ 입력 형식

시 분 초가
시:분:초 형식으로 입력된다.

◎ 출력 형식

분만 출력한다.

입력 예시	출력 예시
17:23:57	23



연월일 입력 받아 형식 바꿔 출력하기

연월일을 표기하는 방법은 나라마다 형식마다 조금씩 다르다.

연월일(yyyy.mm.dd)을 입력받아, 일월연(dd-mm-yyyy) 형식으로 바꿔 출력해보자.
(단, 한 자리 일/월은 0을 붙여 두 자리로, 연도도 0을 붙여 네 자리로 출력해야 한다.)

참고
출력하는 자릿수를 지정하기 위해 %4d와 같은 형식을 사용할 수 있는데,
부족한 자리를 0으로 채우기 위해서는 %04d와 같은 형식을 사용하면 된다.

예시
```
printf("%02d-%02d-%04d", d, m, y);
```

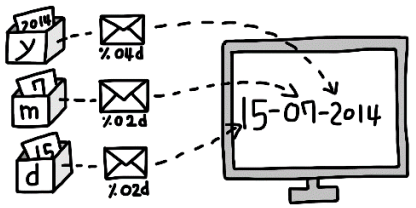
◎ 입력 형식

년월일이 '.'(닷)으로 구분되어 입력된다.

◎ 출력 형식

일월연으로 바꾸어 '-'(대쉬, 마이너스)로 구분해 출력한다.

입력 예시	출력 예시
2014.07.15	15-07-2014



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 한 개 입력받아 그대로 출력하기2

정수 한 개를 입력받아 그대로 출력해보자.
(단, 입력되는 정수의 범위는 0 ~ 4,294,967,295 이다.)

참고
-2147483648 ~ +2147483647 범위의 정수를 저장하고 처리하기 위해서는
int 형으로 변수를 선언하면 된다.(int 로 선언하고 %d로 받고 출력)

하지만 이 범위를 넘어가는 정수를 저장하기 위해서는
보다 큰 범위를 저장할 수 있는 다른 데이터형을 사용해야 정상적으로 저장시킬 수 있다.

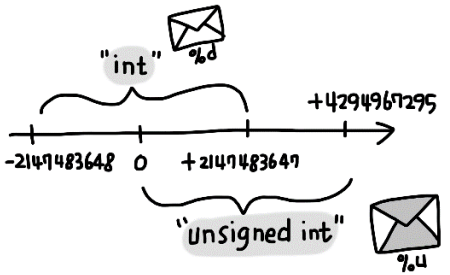
unsigned int 데이터형을 사용하면 0 ~ 4294967295 범위의 정수를 저장할 수 있다.

예시
```
unsigned int n;
scanf("%u", &n);
printf("%u", n);
```

◎ 입력 형식
정수 한 개가 입력된다.
(단, 입력되는 정수의 범위는 0 ~ 4294967295 이다.)

◎ 출력 형식
입력된 정수를 그대로 출력한다.

입력 예시	출력 예시
2147483648	2147483648



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

실수 한 개 입력받아 그대로 출력하기2

실수 한 개를 입력받아 그대로 출력해보자.
(단, 입력되는 실수의 범위는 +- 1.7*10^(-308) ~ +- 1.7*10^(308) 이다.)

참고
float 데이터형을 사용하면 +- 3.4*10^(-38) ~ +- 3.4*10^(38) 범위의 실수를 저장할 수 있다.
(float 로 선언하고 %f로 입력 받아 출력하면 된다.)

이 범위를 넘어가는(더 작거나 더 큰) 실수를 저장하기 위해서는
보다 큰 범위를 저장할 수 있는 다른 데이터형을 사용해야 정상적으로 저장시킬 수 있다.
double은 더 정확하게 저장할 수 있지만, float보다 2배의 저장 공간이 필요하다.

double 데이터형을 사용하면
+- 1.7*10^(-308) ~ +- 1.7*10^(308) 범위의 실수를 저장할 수 있다.

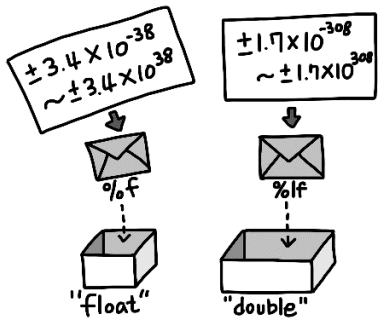
예시
```
double d;
scanf("%lf", &d);  // double(long float) 형식으로 입력
printf("%lf", d);
```

◎ 입력 형식
소수점 아래의 수가 11개 이하인 실수 한 개가 입력된다.
(단, 입력되는 실수의 범위는 +- 1.7*10^(-308) ~ +- 1.7*10^(308) 이다.)

◎ 출력 형식
입력된 실수를 소수점 이하 11자리까지 반올림하여 출력한다.

참고
%.11lf 를 사용하면 소수점 이하 11자리까지 출력된다.

입력 예시	출력 예시
3.14159265359	3.14159265359



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 한 개 입력받아 그대로 출력하기3

정수 한 개를 입력받아 그대로 출력해보자.
단, 입력되는 정수의 범위는
-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807 이다.

참고
-2147483648 ~ +2147483647 범위의 정수를 저장하고 처리하기 위해서는
int 데이터형을 사용해 변수를 선언하면 된다.
(int 로 선언하고 %d로 입력받아 저장하고, 출력하면 된다.)

int 형으로 저장할 수 있는 범위를 넘어가는 정수 값을 저장하기 위해서는
보다 큰 범위를 저장할 수 있는 다른 데이터형을 사용해야 한다.

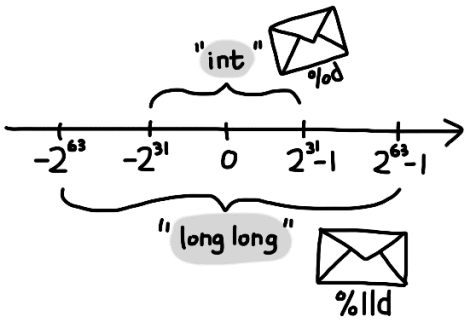
long long int 데이터형을 사용하면
-9,223,372,036,854,775,808 ~ +9,223,372,036,854,775,807 범위의 정수값을
저장시킬 수 있다.

예시
```
long long int n;
scanf("%lld", &n);
printf("%lld", n);
```

◎ 입력 형식
정수 한 개가 입력된다.
단, 입력되는 정수의 범위는 -9223372036854775808 ~ +9223372036854775807 이다.

◎ 출력 형식
입력된 정수를 그대로 출력한다.

입력 예시	출력 예시
-2147483649	-2147483649



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

10진 정수 한 개 입력받아 8진수로 출력하기

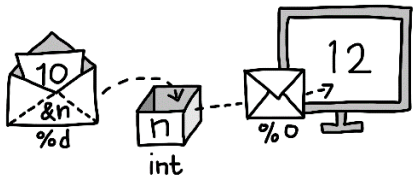
10진수를 입력받아 8진수(octal)로 출력해보자.

참고
%d(10진수 형태)로 입력받고,
%o를 사용해 출력하면 8진수(octal)로 출력된다.

◎ 입력 형식
10진수 1개가 입력된다.
단, 입력되는 정수는 int 범위이다.

◎ 출력 형식
8진수로 출력한다

입력 예시	출력 예시
10	12



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

10진 정수 입력받아 16진수로 출력하기1

10진수를 입력받아 16진수(hexadecimal)로 출력해보자.

참고
%d(10진수 형태)로 입력받고
%x로 출력하면 16진수(hexadecimal) 소문자로 출력된다.

10진법은 한 자리에 10개(0 1 2 3 4 5 6 7 8 9)의 문자를 사용하고,
16진법은 한 자리에 16개(0 1 2 3 4 5 6 7 8 9 a b c d e f)의 문자를 사용한다.
16진수의 a는 10진수의 10, b는 11, c는 12 ... 와 같다.

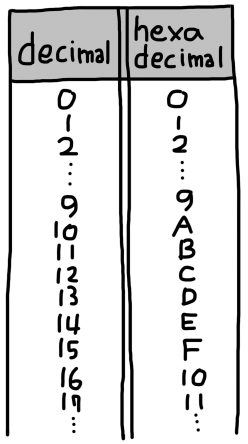
◎ 입력 형식

10진수 1개가 입력된다.

◎ 출력 형식

16진수(소문자)로 출력한다.

입력 예시	출력 예시
255	ff



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

10진 정수 입력받아 16진수로 출력하기2

10진수를 입력받아 16진수(hexadecimal)로 출력해보자.

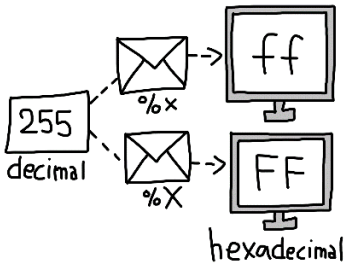
참고
%d(10진수 형태)로 입력받고
%X로 출력하면 16진수(hexadecimal) 대문자로 출력된다.

10진법은 한 자리에 10개(0 1 2 3 4 5 6 7 8 9)의 문자를 사용하고,
16진법은 한 자리에 16개(0 1 2 3 4 5 6 7 8 9 A B C D E F)의 문자를 사용한다.
(알파벳 대소문자는 표현만 다르고 같은 값을 의미한다.)
16진법의 A는 10진법의 10, B는 11, C는 12 ... 와 같다.

◎ 입력 형식
10진수 1개가 입력된다.

◎ 출력 형식
16진수(대문자)로 출력한다.

입력 예시	출력 예시
255	FF



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

8진 정수 한 개 입력받아 10진수로 출력하기

8진수로 입력된 정수 1개를 10진수로 바꾸어 출력해보자.

참고
%o로 입력받으면 8진수로 인식시켜 저장시킬 수 있다.
%d로 출력하면 10진수로 출력된다.

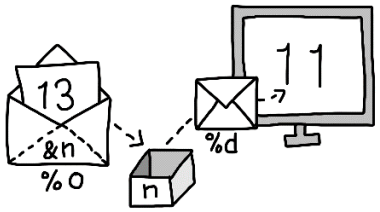
예시
```
int n;
scanf("%o", &n);
printf("%d", n);
```

(C언어에서 소스 코드 작성 시 0으로 시작하는 수는 8진수로 인식된다.
`int a = 013; // 10진수 11과 같은 값`)

◎ 입력 형식
8진 정수 1개가 입력된다.

◎ 출력 형식
10진수로 바꾸어 출력한다.

입력 예시	출력 예시
13	11



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

16진 정수 한 개 입력받아 8진수로 출력하기

16진수로 입력된 정수 1개를 8진수로 바꾸어 출력해보자.

참고
%x(영문자 소문자) 나 %X(영문자 대문자)로 입력 받으면
16진수로 인식시켜 저장시킬 수 있다. %o로 출력하면 8진수로 출력된다.

C언어에서 소스 코드 작성시 0으로 시작하는 수는 8진수로 인식된다.
또한 소스코드 내에서 //로 시작하면 1줄 설명을 넣을 수 있다.
여러 줄을 설명(주석) 을 넣을 경우 /* 와 */ 사이에 작성하면 된다.

예시

```
int n;
scanf("%x", &n); //소문자로 16진수 입력
printf("%o", n);
```

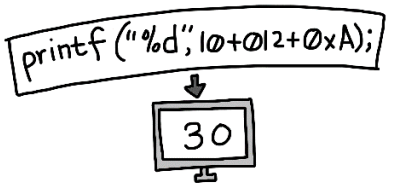
◎ 입력 형식

16진 정수 1개가 입력된다.
(단, 16진수는 영문 소문자로 입력된다.)

◎ 출력 형식

8진수로 바꾸어 출력한다.

입력 예시	출력 예시
f	17



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

영문자 한 개 입력받아 10진수로 출력하기

영문자 한 개를 입력받아 아스키 코드표의 10진수 값으로 출력해보자.

참고
아스키 코드표는
(ASCII, 미국표준코드, American Standard Code for Information Interchange) 영문자, 특수문자 등을 저장할 때 사용하는 표준 코드이다.

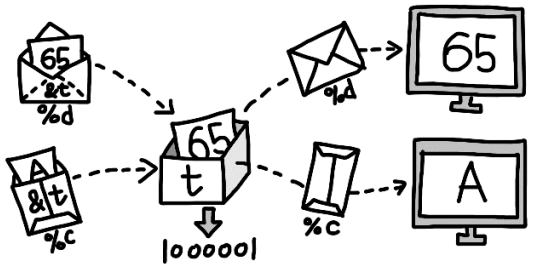
컴퓨터로 저장되는 모든 데이터는 2진 정수화되어 저장되는데,
영문자와 특수기호 등을 저장하는 방법으로 아스키코드가 기본적으로 사용된다.

예를 들어 영문 대문자 'A'는 10진수 65를 의미하는 2진수 값으로 저장된다.

◎ 입력 형식
영문자 1개가 입력된다.

◎ 출력 형식
아스키코드 값을 10진수로 출력한다.

입력 예시	출력 예시
A	65



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 입력받아 아스키 문자로 출력하기

10진 정수 한 개를 입력받아 아스키 문자로 출력해보자.
단, 0 ~ 255 범위의 정수만 입력된다.

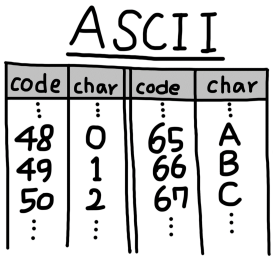
◎ 입력 형식

10진 정수 1개(0 ~ 255 범위)가 입력된다.

◎ 출력 형식

아스키코드 값을 문자로 출력한다.

입력 예시	출력 예시
65	A



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 두 개 입력받아 합 출력하기1

정수 2개를 입력받아 합을 출력해보자.
단, 입력되는 정수는 -1073741824 ~ +1073741824 이다.

참고
+ 연산자를 사용하면 된다.
일반적인 사칙연산을 위한 연산자는 +, -, *, / 를 사용한다.

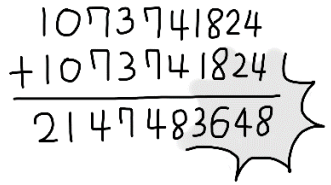
하지만, 나누기 연산자인 '/' 는 약간 다른 의미를 가진다.

계산된 결과가 int 범위를 넘어간다면 다른 데이터형을 사용해야 한다.

◎ 입력 형식
두 개의 정수가 공백으로 구분되어 입력된다.
** 주의 : 계산의 결과가 int 범위를 넘어가는지를 잘 생각해 보아야 한다.

◎ 출력 형식
두 정수의 합을 출력한다.

입력 예시	출력 예시
123 -123	0



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 두 개 입력받아 합 출력하기2

정수 2개를 입력받아 합을 출력해보자.
단, 입력되는 정수는 -2147483648 ~ +2147483648 이다.

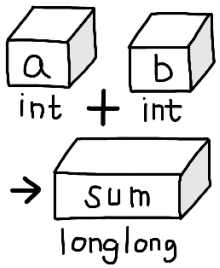
참고
+ 연산자를 사용하면 된다.
단, 계산된 결과가 int 형으로 저장할 수 있는 범위를 넘어갈 수 있기 때문에 다른 데이터형을 사용해야 한다.

주의
int 데이터형은 %d로 입출력하고,
long long int 데이터형은 %lld로 입출력한다.

◎ 입력 형식
두 개의 정수가 공백으로 구분되어 입력된다.

◎ 출력 형식
두 정수의 합을 출력한다.

입력 예시	출력 예시
2147483648 2147483648	4294967296



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 한 개 입력받아 부호 바꿔 출력하기

입력된 정수의 부호를 바꿔 출력해보자.
(단, -2147483647 ~ +2147483647 범위의 정수가 입력된다.)

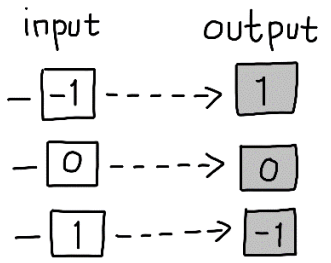
참고
단항 연산자인 -(negative)를 변수 앞에 붙이면 부호가 반대로 바뀌어 계산된다.

예시
```
int a;
scanf("%d", &a);
printf("%d", -a);
```

◎ 입력 형식
정수 한 개가 입력된다.

◎ 출력 형식
부호를 바꿔 출력한다.

입력 예시1	출력 예시1	입력 예시2	출력 예시2
-1	1	0	0



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

문자 한 개 입력받아 다음 문자 출력하기

영문자 한 개를 입력받아 그 다음 문자를 출력해보자.
영문자 'A'의 다음 문자는 'B'이고, 영문자 '0'의 다음 문자는 '1'이다.

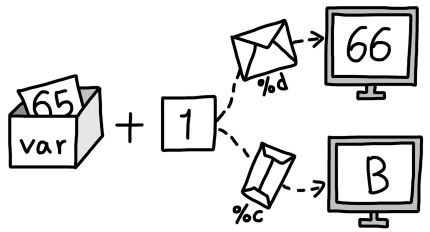
참고
숫자는 수를 표현하는 문자로서 '0' 은 문자 그 자체를 의미하고, 0은 값을 의미한다.

힌트
아스키문자표에서 'A'는 10진수 65로 저장되고 'B'는 10진수 66으로 저장된다.
따라서 문자도 값으로 덧셈을 할 수 있다.

◎ 입력 형식
영문자 한 개가 입력된다.

◎ 출력 형식
다음 문자를 출력한다.

입력 예시	출력 예시
a	b



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 두 개 입력받아 나눈 몫 출력하기

정수 두 개(a, b) 를 입력받아 a를 b로 나눈 몫을 출력해보자.
단, -2147483648 <= a <= b <= +2147483647, b는 0이 아니다.

참고
C언어에서 정수/정수 연산의 결과는 정수(몫)로 계산된다.

실수/정수, 정수/실수, (float)정수/정수, 정수/(float)정수 등의 연산 결과는
실수 값으로 계산된다.

(float)(정수/정수)는 정수/정수의 결과인 정수 값을 실수형(float)로 바꾸는 것으로,
계산 결과인 정수 값이 실수 형태로 변환되는 것이니 주의해야 한다.
정수/실수 계산결과가 자동으로 실수형으로 바뀌는 것을 묵시적 (데이터)형변환이라고 하고,
어떤 값이나 결과의 데이터형을 강제로 바꾸는 것을 명시적 (데이터)형변환이라고 한다.

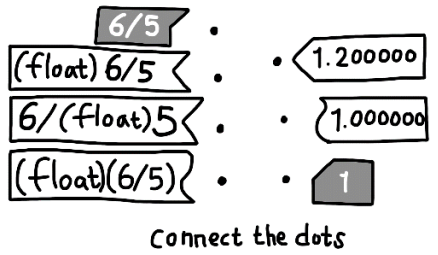
```
int a;
scanf("%d", &a);
printf("%f", (float)a);
```

와 같이 실행하면,
정수로 저장되어 있는 값을 실수형(float)으로 명시적으로 변환하여 출력한다.

◎ 입력 형식
정수 두 개(a, b)가 공백을 두고 입력된다.
단, -2147483648 <= a <= b <= +2147483647

◎ 출력 형식
a를 b로 나눈 몫을 출력한다.

입력 예시1	출력 예시1	입력 예시2	출력 예시2
1 3	0	10 3	3



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 두 개 입력받아 나눈 나머지 출력하기

정수 두 개(a, b) 를 입력받아 a를 b로 나눈 나머지를 출력해보자.
단, 0 <= a, b <= +2147483647, b는 0이 아니다.

참고
C언어에서 정수%정수 연산의 결과는 나눈 나머지로 계산된다.

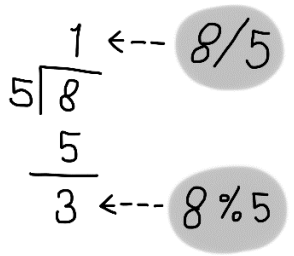
% 연산자(modulus, mod 연산) 수학자 가우스가 생각해낸 연산으로,
어떤 정수를 다른 정수로 나누고 난 후 남는 나머지를 계산하는 연산이다.

단, 음(-)이 아닌 정수에 대해서만 연산된다.

◎ 입력 형식
정수 두 개(a, b)가 공백을 두고 입력된다.
단, 0 <= a, b <= +2147483647, b는 0이 아니다.

◎ 출력 형식
a 를 b로 나눈 나머지를 출력한다.

입력 예시1	출력 예시1	입력 예시2	출력 예시2
10 3	1	3 7	3



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 한 개 입력받아 1 더해 출력하기

정수를 한 개 입력받아 1만큼 더해 출력해보자.
단, -2147483648 ~ +2147483647 의 범위로 입력된다.

주의
계산되고 난 후의 값의 범위(데이터형)에 주의한다.

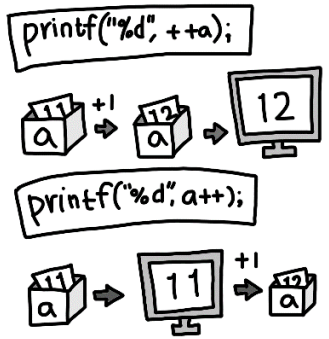
참고
어떤 변수(a)에 값을 저장한 후 a+1 의 값을 출력할 수도 있고,
++a 연산을 한 후에 출력할 수도 있다.
++a, --a, a++, a-- 와 같이 어떤 변수의 앞이나 뒤에 붙여
변수에 저장되어있는 값을 1만큼 더하거나 빼주는 연산자를 증감연산자라고 한다.

증감연산자를 변수 앞에 붙이면 그 변수를 사용하기 전에 증감을 먼저 수행하고,
증감연산자를 변수 뒤에 붙이면 일단 변수에 저장되어있는 값을 먼저 사용하고 난 후
나중에 증감을 수행한다.

◎ 입력 형식
정수 한 개가 입력된다.
단, -2147483648 ~ +2147483647 의 범위로 입력된다.

◎ 출력 형식
입력된 정수에 1을 더해 출력한다.

입력 예시1	출력 예시1	입력 예시2	출력 예시2
123	124	2147483647	2147483648



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 두 개 입력받아 자동 계산하기

정수 두 개(a, b)를 입력받아 합, 차, 곱, 몫, 나머지, 나눈 값을 자동으로 계산해보자.
단 0 <= a, b <= 2147483647, b는 0이 아니다.

◎ 입력 형식

정수 두 개가 공백을 두고 입력된다.

◎ 출력 형식

첫 줄에 합
둘째 줄에 차,
셋째 줄에 곱,
넷째 줄에 몫,
다섯째 줄에 나머지,
여섯째 줄에 나눈 값을 순서대로 출력한다.
(실수, 소수점 이하 셋째 자리에서 반올림해 둘째 자리까지 출력)

입력 예시	출력 예시
10 3	13 7 30 3 1 3.33

Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

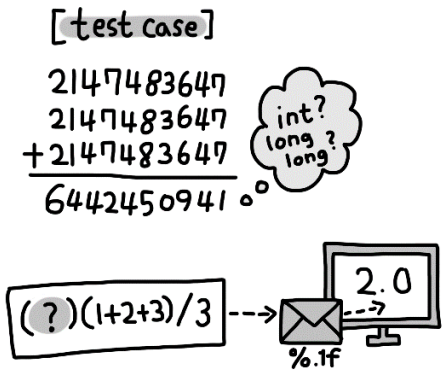
정수 세 개 입력받아 합과 평균 출력하기

정수 세 개를 입력받아 합과 평균을 출력해보자.
단, -2147483648 ~ +2147483647

◎ 입력 형식
정수 세 개가 공백을 두고 입력된다.
단, -2147483648 ~ +2147483647

◎ 출력 형식
합과 평균을 줄을 바꿔 출력한다.
평균은 소수점 이하 둘째 자리에서 반올림해서 소수점 이하 첫째 자리까지 출력한다.

입력 예시	출력 예시
1 2 3	6 2.0



정수 한 개 입력받아 2배 곱해 출력하기

정수 한 개를 입력받아 2배 곱해 출력해보자.

참고
*2 의 값을 출력해도 되지만,
정수를 2배로 곱하거나 나누어 계산해 주는 비트단위시프트연산자 <<, >>를 이용한다.

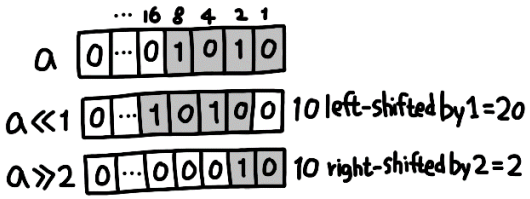
2진수 형태로 저장되어 있는 값들을 왼쪽(<<)이나 오른쪽(>>)으로
지정한 비트 수만큼 밀어주면 2배씩 늘어나거나 반으로 줄어드는데,

왼쪽 비트시프트(<<)가 될 때에는 오른쪽에 0이 주어진 개수만큼 추가되고,

오른쪽 비트시프트(>>)가 될 때에는
왼쪽에 0(0 또는 양의 정수인 경우)이나 1(음의 정수인 경우)이 개수만큼 추가된다.

범위(32비트)를 넘어서 이동되는 비트는 삭제된다.

예시
```
int a=10;
printf("%d", a<<1); //10을 2배 한 값인 20 이 출력된다.
printf("%d", a>>1); //10을 반으로 나눈 값인 5 가 출력된다.
printf("%d", a<<2); //10을 4배 한 값인 40 이 출력된다.
printf("%d", a>>2); //10을 반으로 나눈 후 다시 반으로 나눈 값인 2 가 출력된다.
```



◎ 입력 형식
정수 한 개가 입력된다.
단, -1073741824 ~ +1073741823

◎ 출력 형식
2배 곱한 정수를 출력한다.

입력 예시	출력 예시
1024	2048

Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

한 번에 2^n 배로 출력하기

정수 두 개(a, b)를 입력받아 a를 2^b배 곱한 값으로 출력해보자.
0 <= a <= 10, 0 <= b <= 10

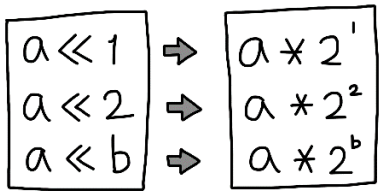
참고
예를 들어 1 3 이 입력되면 1을 2^3(8)배 하여 출력한다.

예시
```
int a=1, b=10;
printf("%d", a << b); //2^10 = 1024 가 출력된다.
```

◎ 입력 형식
정수 두 개가 공백을 두고 입력된다.
0 <= a, b <= 10

◎ 출력 형식
a 를 2^b배 만큼 곱한 값을 출력한다.

입력 예시	출력 예시
1 3	8



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

두 정수 입력받아 비교하기1

두 정수(a, b)를 입력받아

a가 b보다 크면 1을, a가 b보다 작거나 같으면 0을 출력하는 프로그램을 작성해보자.

참고
어떤 값을 비교하기 위해 비교/관계연산자(comparison/relational)를 사용할 수 있다.

비교/관계연산자 > 는
왼쪽의 값이 오른쪽 값 보다 큰 경우 참(true)을 나타내는 정수값 1로 계산하고,
그 외의 경우에는 거짓(false)를 나타내는 정수값 0으로 계산한다.

비교/관계연산자도 일반적인 사칙연산자처럼 주어진 두 수를 이용해 계산을 수행하고,
그 결과를 1(참), 또는 0(거짓)으로 계산해 주는 연산자이다.

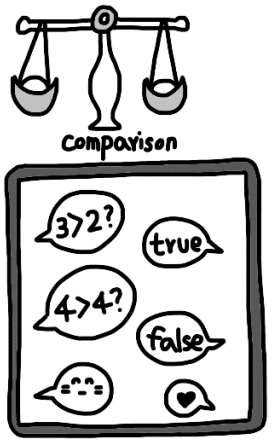
비교/관계연산자는 >, <, >=, <=, ==(같다), !=(다르다) 의 6가지가 있다.

예시
```
printf("%d", 123<456); //비교 연산자 < 의 계산 결과인 1(참)이 출력된다.
```

◎ 입력 형식
두 정수 a, b가 공백을 두고 입력된다.
-2147483648 <= a, b <= +2147483647

◎ 출력 형식
a가 b보다 큰 경우 1을, 그렇지 않은 경우 0을 출력한다.

입력 예시1	출력 예시1	입력 예시2	출력 예시2
9 1	1	123 456	0



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

두 정수 입력받아 비교하기2

두 정수(a, b)를 입력받아
a와 b가 같으면 1을, 같지 않으면 0을 출력하는 프로그램을 작성해보자.

참고
어떤 값을 비교하기 위해 비교/관계연산자(comparison/relational)를 사용할 수 있다.

비교/관계연산자 == 는
두 개의 값이 같은 경우 참(true)을 나타내는 정수값 1로 계산하고,
다른 경우 거짓(false)를 나타내는 정수값 0으로 계산한다.

비교/관계연산자도 일반적인 사칙연산자처럼 주어진 두 수를 이용해 계산을 수행하고,
그 결과를 1(참), 또는 0(거짓)으로 계산해 주는 연산자이다.

비교/관계연산자는 >, <, >=, <=, ==(같다), !=(다르다)의 6가지가 있다.

** 수학에서 왼쪽과 오른쪽의 계산 결과가 같음(동치)을 나타내는 기호 = 는
C언어에서 전혀 다른 의미로 사용된다.

a=1 와 같은 표현은 a와 1의 값이 같다는 의미가 아니라
오른쪽의 계산 결과인 1을 왼쪽의 변수 a에 저장하라는 의미이다.

◎ 입력 형식
두 정수 a, b가 공백을 두고 입력된다.
-2147483648 <= a, b <= +2147483647

◎ 출력 형식
a와 b의 값이 같은 경우 1을, 그렇지 않은 경우 0을 출력한다.

입력 예시1	출력 예시1	입력 예시2	출력 예시2
0 0	1	1 999	0