문자 한 개 입력받아 알파벳 출력하기

영문자(a ~ z) 한 개가 입력되었을 때 그 문자까지의 알파벳을 순서대로 출력해보자.

참고
```
do
{             //코드블록
 ...
}while(조건);  //do~while( ) ; 구조는 반드시 마지막에 세미콜론(;)을 붙여야 한다.
```

구조를 사용하자.

```
do
{             //코드블록
 ...
}while(조건);
```
구조는 while( ) { ... } 과 유사한 반복 실행 구조를 만들 수 있는데
다른 점은 무조건 한 번은 실행된다는 것이다. 마지막에 세미콜론을 반드시 붙여야 한다.

예시
```
char x, t='a';
scanf("%c", &x);
do
{
  printf("%c ", t);
  t+=1;             //t+=1; 는 t=t+1과 같은 의미이다.
}while(t<'z'+1);    //의미 : t의 값이 'z'보다 작은 동안 반복된다.
```

** 복합 대입 연산자에 대해...
컴퓨터의 세상에서 반복되는 것은 다시 쓰기 귀찮다.
즉, 어딘가에 정의가 되어있다면 재사용하던가, 아니면 보다 간단히 쓰고 싶어한다.

복합 대입 연산자는 대입 연산자를 함께 사용해 보다 간단히 표현하는 방법이다.
예를 들어 어떤 연산자 X와 대입 연산자를 합쳐 "X="라고 표현하면,
이는 X 연산 후 대입하라는 의미를 간단히 나타내는 것이다.

+=, -=, *=, /=, %= ... 등의 형태들이 모두 가능하며 의미는 아래와 같다.

예를 들어

```
n = n + 3;
```
은
n += 3; 으로 간단히 표현할 수 있다.

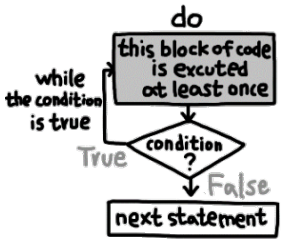
```
n = n << 1;
```
는
n <<= 1; 로 간단히 표현할 수 있다.

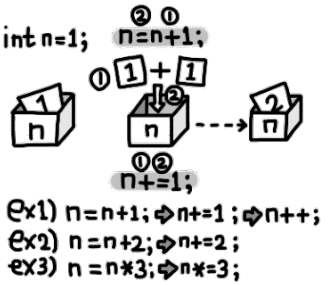
아래는 서로 다른 형태이지만 같은 기능을 하는 코드들이다.
```
n++;
n=n+1;
n+=1;
```

◎ 입력 형식
영문자 한 개가 입력된다.
(a ~ z)

◎ 출력 형식
a부터 입력한 문자까지 순서대로 공백을 두고 출력한다.

입력 예시	출력 예시
f	a b c d e f





Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

정수 한 개 입력받아 그 수까지 출력하기

정수(0 ~ 100) 한 개를 입력받아 0부터 그 수까지 순서대로 출력해보자.

for((반복전 실행/준비 내용); (조건 검사 내용); (한번 실행한 후 처리할 작업))
{ //코드블록
  ...
}
반복 구조를 사용하자.

참고
for((반복전 실행/준비내용); (조건 검사 내용); (한번 실행한 후 처리할 작업))
{ //코드블록
  ...
}
구조는 while( ) {...}, do {...} while( ); 구조와 같이
어떤 작업을 반복적으로 수행할 때 가장 일반적으로 많이 사용하는 구조이다.

while( ) {...}, do {...} while( ); 과 다르게
반복을 위한 초기화/준비, 조건검사, 반복 후 증감 등을 한 번에 작성할 수 있다.

for(...) 를 사용할 때에는 반복 실행되는 과정에 대해 정확히 이해할 필요가 있는데.
1. 반복 전 실행 수행(1번만 실행된다.)
2. 조건검사
3. 조건이 참이면 코드블록 실행
4. 코드블록 실행 후 증감 또는 처리 : 한 번 반복 후 처리
5. 조건검사
6. 조건이 참이면 코드블록 실행
...
의 순서로 처리된다.

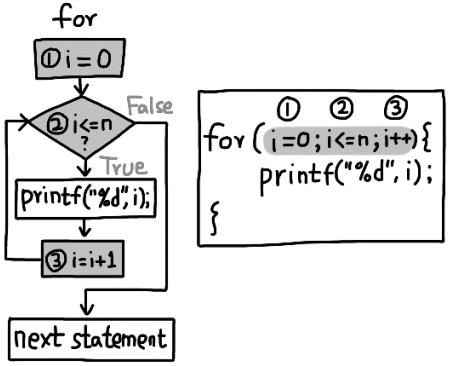
예시
```
int i, n;
scanf("%d", &n);
for(i=0; i<=n; i++)
{
  printf("%d ");
}
printf("\n");
```

** 일반적으로 반복 횟수를 카운트하면서 비교하기 위해 i 변수를 많이 사용한다.
   i : index, iterator ?

◎ 입력 형식
정수 한 개가 입력된다.
(0 ~ 100)

◎ 출력 형식
0부터 그 수까지 줄을 바꿔 한 개씩 출력한다.

입력 예시	출력 예시
4	0 1 2 3 4



짝수 합 구하기

정수(1 ~ 100) 한 개를 입력받아 1 부터 그 수까지 짝수의 합을 구해보자.

참고
while( ) {...}, do {...} while( );, for(...; ...; ...) {...} 등의 반복문은 형태만 다르고, 똑같은 성능을 발휘한다. 필요에 따라 편리한 것으로 골라 사용하면 된다.

예시
```
//아래의 예는 홀수일 경우만 더해 그 결과를 출력한다.
int sum=0;
int i, n;
scanf("%d", &n);
for(i=1; i<=n; i++)
{
  if(i%2==1) sum=sum+i;
}
```

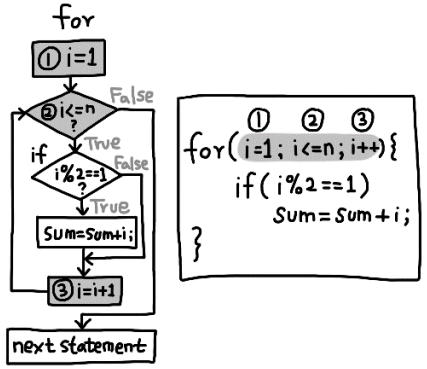
◎ 입력 형식

정수 한 개가 입력된다.
(1 ~ 100)

◎ 출력 형식

1부터 입력된 수까지 짝수의 합을 출력한다.

입력 예시	출력 예시
5	6



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

원하는 문자가 입력될 때까지 반복 출력하기

q 가 입력될 때까지 입력한 문자를 계속 출력하는 프로그램을 작성해보자.

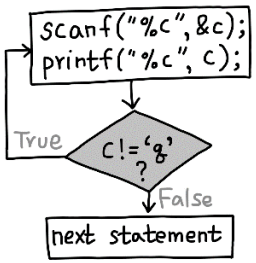
◎ 입력 형식

문자 한 개가 계속해서 입력된다.

◎ 출력 형식

q 가 입력될 때까지 입력된 문자를 줄을 바꿔 한 줄씩 출력한다.

입력 예시	출력 예시
x b k d l q	x b k d l q



언제까지 더해야 할까?

1, 2, 3 ... 을 계속 더해 나갈 때,
그 합이 입력한 정수(0 ~ 1000)보다 같거나 작을 때까지
계속 더하는 프로그램을 작성해보자.

즉, 1부터 n까지 정수를 계속 더한다고 할 때,
어디까지 더해야 입력한 수보다 같거나 커지는지 알아보고자 하는 문제이다.

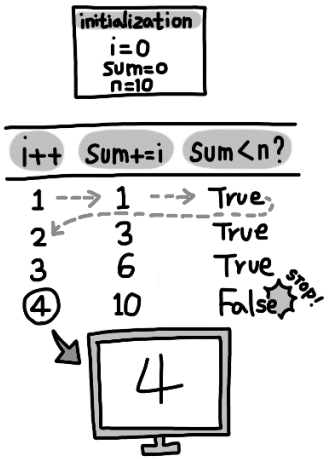
◎ 입력 형식

정수 한 개가 입력된다.

◎ 출력 형식

1, 2, 3, 4, 5 ... 를 계속 더해가다가,
입력된 정수와 같거나 커졌을 때, 마지막에 더한 정수를 출력한다.

입력 예시	출력 예시
55	10



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

주사위를 2개 던지면?

1부터 n까지, 1부터 m까지 숫자가 적힌
서로 다른 주사위 2개를 던졌을 때 나올 수 있는 모든 경우를 출력해보자.

참고
중첩의 원리...
반복 실행 구조도 조건 실행 구조와 마찬가지로 중첩의 원리가 적용된다.

반복 실행 구조를 중첩하면 원하는 반복 구조를 다양하게 만들어 낼 수 있다.

예시

```
int i, j;
int n=3, m=6;
for(i=1; i<=n; i++)
{
  for(j=1; j<=m; j++)
  {
    printf("%d %d ", i, j);
  }
}
```

의 코드는
바깥쪽의 i가 1부터 n까지로 바뀌는 각각의 동안에
안쪽의 j가 1부터 m까지 변하며 출력된다.

중첩 반복되는 코드블록의 내용이 논리적으로 한 개의 단위이기 때문에
코드블록 기호를 생략하면 아래와 같이 작성할 수도 있다.

```
int i, j;
int n=3, m=6;
for(i=1; i<=n; i++)
  for(j=1; j<=m; j++)
    printf("%d %d ", i, j);
```

반복 실행구조를 사용할 때에는
반복횟수를 기록해 두기 위해 i, j와 같은 변수를 선언해 사용해야 한다.

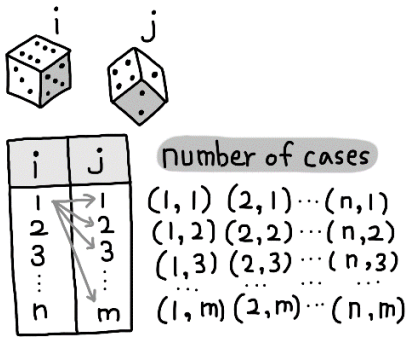
◎ 입력 형식

서로 다른 주사위의 면의 개수 n, m이 공백을 두고 입력된다.
단, n, m은 10이하의 자연수

◎ 출력 형식

나올 수 있는 주사위의 숫자를 한 세트씩 줄을 바꿔 모두 출력한다.
첫 번째 수는 n, 두 번째 수는 m으로 고정해 출력하도록 한다.

입력 예시	출력 예시
2 3	1 1
	1 2
	1 3
	2 1
	2 2
	2 3



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

16진수 구구단?

16진수(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)를 배운
영일(01)이는 16진수끼리 곱하는 16진수 구구단?에 대해서 궁금해졌다.

A, B, C, D, E, F 중 하나가 입력될 때,
1부터 F까지 곱한 16진수 구구단의 내용을 출력해보자.
(단, A ~ F 까지만 입력된다.)

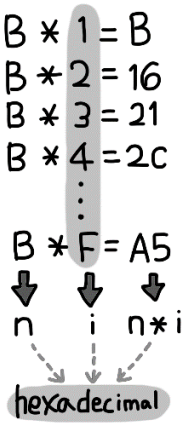
◎ 입력 형식

16진수로 한 자리 수가 입력된다.
단, A ~ F 까지만 입력된다.

◎ 출력 형식

입력된 16진수에 1~F까지 순서대로 곱한, 16진수 구구단을 줄을 바꿔 출력한다.
계산 결과도 16진수로 출력해야 한다.

입력 예시	출력 예시
B	B*1=B
	B*2=16
	B*3=21
	B*4=2C
	B*5=37
	B*6=42
	B*7=4D
	B*8=58
	B*9=63
	B*A=6E
	B*B=79
	B*C=84
	B*D=8F
	B*E=9A
	B*F=A5



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

3 6 9 게임의 왕이 되자!

3 6 9 게임을 하던 영일이는 3 6 9 게임에서 잦은 실수로 계속해서 벌칙을 받게 되었다.
3 6 9 게임의 왕이 되기 위한 마스터 프로그램을 작성해 보자.

참고
중첩의 원리
반복 실행 구조 안에 선택 실행 구조를 자유롭게 중첩할 수 있다.

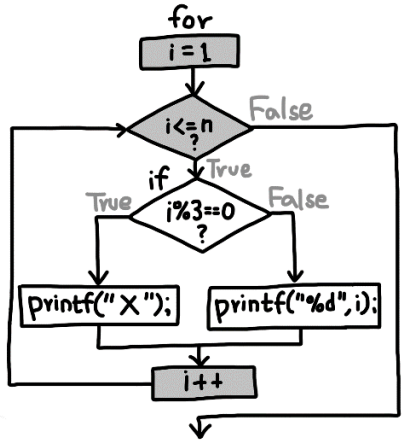
예시
```
int i;
for(i=1; i<=100; i++) //1부터 100까지 반복
{
  if(i%3==0 || i%5==0) printf("%d ", i); //3또는 5의 배수인 경우 그 수 출력
}
```

** 3 6 9 게임은?
여러 사람이 순서를 정해 순서대로 수를 부르는 게임이다.
만약 3, 6, 9 가 들어간 수를 자신이 불러야 하는 상황이면, 대신 "박수" 를 쳐야 한다.
33까지 진행했다면? "짝짝"과 같이 박수를 두 번 치는 형태도 있다.

◎ 입력 형식
10보다 작은 정수 한 개가 입력된다.
1 ~ 10

◎ 출력 형식
1 부터 그 수까지 순서대로 공백을 두고 수를 출력하는데,
3 또는 6 또는 9인 경우 그 수 대신 영문 대문자 X 를 출력한다.

입력 예시	출력 예시
9	1 2 X 4 5 X 7 8 X



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

빛 섞어 만들기

빨강(red), 초록(green), 파랑(blue) 빛을 섞어
여러 가지 빛의 색을 만들어 내려고 한다.

빨강(r), 초록(g), 파랑(b) 각각의 빛의 개수가 주어질 때,
(빛의 강약에 따라 0 ~ n-1 까지 n가지의 빛 색깔을 만들 수 있다.)

주어진 rgb 빛들을 다르게 섞어 만들 수 있는 모든 경우의 조합(r g b)과
총 가짓 수를 계산해보자.

예시

```
int i, j, k, c=0;
int r, g, b;
scanf("%d%d%d", &r, &g, &b);

for(i=0; i<r; i++)
  for(j=0; j<g; j++)
    for(k=0; k<b; k++)
    {
      printf("%d %d %d\n", i, j, k);
      c++;
    }

printf("%d ", c);
```

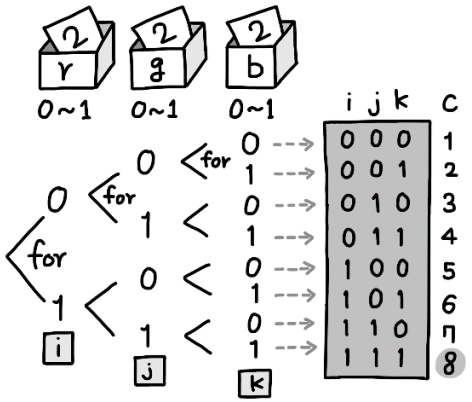
◎ 입력 형식

빨녹파(r, g, b) 각 빛의 강약에 따른 가짓수(0 ~ 128))가 공백을 사이에 두고 입력된다.
예를 들어 3 3 3 은 각 색깔 빛에 대해서 0~2까지 3가지씩 있음을 의미한다.

◎ 출력 형식

만들 수 있는 rgb 빛의 정보를 오름차순(계단을 올라가는 순, 12345... abcde..., 가나다라마...)으로 줄을 바꿔 모두 출력하고, 마지막에 그 개수를 출력한다.

입력 예시	출력 예시
2 2 2	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 8



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)

소리 파일 저장용량 계산하기

소리가 컴퓨터에 저장될 때에는 디지털 데이터화 되어 저장된다.

마이크를 통해 1초에 적게는 수십 번, 많게는 수만 번 소리의 강약을 체크해
그 값을 정수값으로 바꾸고, 그 값을 저장해 소리를 파일로 저장할 수 있다.

값을 저장할 때에는 비트를 사용하는 정도에 따라 세세한 녹음 정도를 결정할 수 있고,
좌우(스테레오) 채널로 저장하면 2배… 5.1채널이면 6배의 저장공간이 필요하고,
녹음 시간이 길면 그 만큼 더 많은 저장공간이 필요하다.

1초 동안 마이크로 소리강약을 체크하는 수를 h
(헤르쯔, Hz 는 1초에 몇 번? 체크하는가를 의미한다.)

한 번 체크한 결과를 저장하는 비트 b
(2비트를 사용하면 0 또는 1 두 가지, 16비트를 사용하면 65536가지..)

좌우 등 소리를 저장할 트랙 개수인 채널 c
(모노는 1개, 스테레오는 2개의 트랙으로 저장함을 의미한다.)

녹음할 시간 s가 주어질 때,

필요한 저장 용량을 계산하는 프로그램을 작성해보자.

실제로 일반적인 CD 음질(44.1KHz, 16bit, 스테레오)로 1초 동안 저장하려면
44100 * 16 * 2 * 1 bit의 저장공간이 필요하다.

이렇게 녹음하는 방식을 PCM(Pulse Code Modulation) 방법이라고 하는데,
압축하지 않은 순수한(raw) 소리 데이터 파일은 대표적으로 *.wav 가 있다.

**
```
   8 bit(비트)       = 1byte(바이트)    // 8bit=1Byte
1024 Byte(2^10 byte) = 1KB(킬로 바이트)
1024 KB(2^10 KB)     = 1MB(메가 바이트)
1024 MB(2^10 MB)     = 1GB(기가 바이트)
1024 GB(2^10 GB)     = 1TB(테라 바이트)
....
```

◎ 입력 형식

h, b, c, s 가 공백을 두고 입력된다.
h는 48,000이하, b는 32이하(단, 8의배수), c는 5이하, s는 6,000이하의 자연수이다.

◎ 출력 형식

필요한 저장 공간을 MB 단위로 바꾸어 출력한다.
단, 소수점 둘째 자리에서 반올림해 첫째 자리까지 출력하고 MB를 공백을 두고 출력한다.

입력 예시	출력 예시
44100 16 2 10	1.7 MB

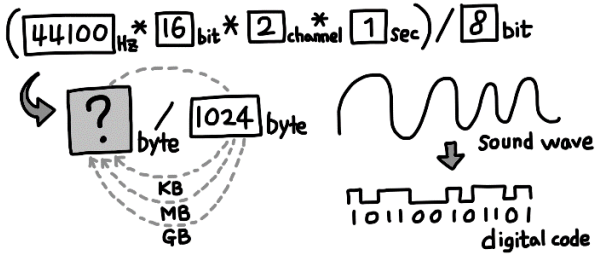


그림 파일 저장용량 계산하기

이미지가 컴퓨터에 저장될 때에도 디지털 데이터화 되어 저장된다.

가장 기본적인 방법으로는 그림을 구성하는 한 점(pixel, 픽셀)의 색상을
빨강(r), 초록(g), 파랑(b)의 3가지의 빛의 세기 값으로 따로 변환하여 저장하는 것인데,

예를 들어 r, g, b 각 색에 대해서 8비트(0~255, 256가지 가능)씩을 사용한다고 하면,

한 점의 색상은 3가지 r, g, b의 8비트+8비트+8비트로 총 24비트로 표현해서
총 2^24 가지의 서로 다른 빛의 색깔을 사용할 수 있는 것이다.

그렇게 저장하는 점을 모아 하나의 큰 이미지를 저장할 수 있게 되는데,
1024 * 768 사이즈에 각 점에 대해 24비트로 저장하면 그 이미지를 저장하기 위한
저장 용량을 계산할 수 있다.

이렇게 이미지의 원래(raw) 데이터를 압축하지 않고 그대로 저장하는 대표적인 이미지 파일이
*.bmp 파일이며, 비트로 그림을 구성한다고 하여 비트맵 방식 또는 래스터 방식이라고 한다.

이미지의 가로 해상도 w, 세로 해상도 h, 한 픽셀을 저장하기 위한 비트 b 가 주어질 때,
압축하지 않고 저장하기 위해 필요한 저장 용량을 계산하는 프로그램을 작성해 보자.

예를 들어
일반적인 1024 * 768 사이즈(해상도)의 각점에 대해
24비트(rgb 각각 8비트씩 3개)로 저장하려면 1024 * 768 * 24 bit의 저장 용량이 필요하다.

실제 그런지 알고 싶다면, 간단한 그림 편집/수정 프로그램을 통해 확인할 수 있다.

**
```
   8 bit(비트)       = 1byte(바이트)    // 8bit=1Byte
1024 Byte(2^10 byte) = 1KB(킬로 바이트)
1024 KB(2^10 KB)     = 1MB(메가 바이트)
1024 MB(2^10 MB)     = 1GB(기가 바이트)
1024 GB(2^10 GB)     = 1TB(테라 바이트)
....
```

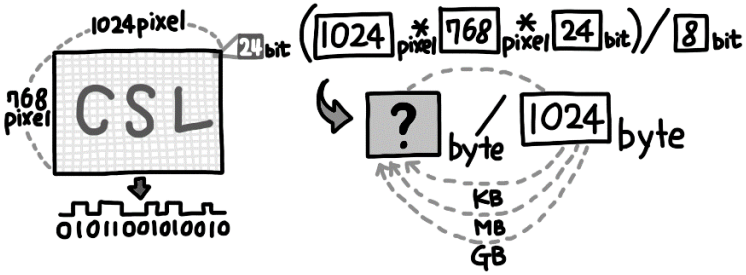
◎ 입력 형식

w, h, b 가 공백을 두고 입력된다.
단, w, h는 모두 정수이고 1~1024 이다. b는 40이하의 4의 배수이다.

◎ 출력 형식

필요한 저장 공간을 MB 단위로 바꾸어 출력한다.
소수점 이하 셋째 자리에서 반올림해 둘째 자리까지 출력한 뒤 MB를 출력한다.

입력 예시	출력 예시
1024 768 24	2.25 MB



여기까지! 이제 그만~

1, 2, 3 ... 을 계속 더해나갈 때,
그 합이 입력한 정수보다 작을 동안(0 ~ 1000) 계속 더하는 프로그램을 작성해보자.

즉, 1부터 n까지 정수를 계속 더한다고 할 때,
어디까지 더해야 입력한 수보다 같거나 커지는지 알아보고자 하는 문제이다.

하지만, 이번에는 그 때의 합을 출력해야 한다.

예를 들어 57을 입력하면
1+2+3+...+8+9+10=55에 다시 11을 더해 66이 될 때,
그 값 66이 출력되어야 한다.

참고
조건문이나 반복문의 코드블록 안에서 break;가 실행되면
가장 가까운 반복 코드블록 구역의 밖으로 빠져나간다.

예시

```
int n, i, s=0;
scanf("%d", &n);
for(i=1; ; i++)   //for 반목문에서 가운데의 조건이 빠진 경우 무한 반복된다.
{
  s+=i;
  if(s>=n)
    break;        //참이면, 가장 가까운 반복 코드블록의 밖으로 빠져나간다.
}                 //break; 가 실행되면 반복을 중단하고 여기로 빠져 나온다.
printf("%d", s);
```

무한 반복이 되는 코드는
while(1) {...}, do {...}while(1); 등도 가능하다.
0이 아니면 모두 참(true)으로 인식되기 때문이다.

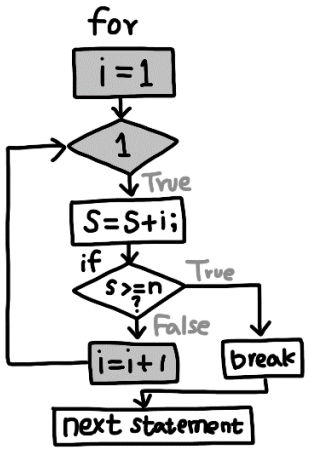
◎ 입력 형식

어느 정도까지 합을 계산할 지, 정수 한 개를 입력받는다.
단, 입력되는 자연수는 100,000,000이하이다.

◎ 출력 형식

1, 2, 3, 4, 5 ... 계속 더해가다가,입력된 정수보다 커지거나 같아지는 경우,
그때까지의 합을 출력한다.

입력 예시	출력 예시
57	66



Thought && (Idea ¦¦ Curiosity)

(Offline Coding) ¦¦ (Kernel Codes)