더블클릭 또는 Enter 키를 눌러 수정

```
a=1000
print(a)
a= -7
print(a)

a=1e9
print(a)
```

```
1000
-7
1000000000.0
```

```
a=round(a,1)0.3+0.6
print()
if a==0.9:
  print(True)
else:
  print(False)
```

```
0.9
False
```

그리디 알고리즘

예제 3-1. 거스름돈

```
n=1260
count=0

coin_types=[500,100,50,10]

for coin in coin_types:
  count+=n//coin # //는 몫을 뜻함
  n%= coin # n %coin: n을 coin으로 나눈 나머지를 n에 저장한다.

print(count)
```

```
6
```

3-2) 큰 수의 법칙

동빈이의 큰 수의 법칙은 다양한 수로 이루어진 배열이 있을 때 주어진 수들을 M번 더하여 가장 큰 수를 만드는 법칙이다. 단 배열의 특정한 인덱스(번호)에 해당하는 수가 연속해서 K번을 초과하여 더해질 수 없는 것이 이 법칙의 특징이다. 예를 들어 순서대로 2,4,5,4,6 으로 이루어진 배열이 있을 때 M이 8이고, K가 3이라고 가정하자. 이 경우 특정한 인덱스의 수가 연속해서 세 번까지만 더해질 수 있으므로 큰 수의 법칙에 따르 결과는 66656665 더한 46이 된다. 단 서로 다른 인덱스에 해당하는 수가 같은 경우에도 서로 다른 것으로 간주한다. 예를 들어 순서대로 34343으로 이루어진 배열이 있을 때 M이 7이고, K가 2라고 가정하자. 이 경우 두 번째 원소에 해당하는 4와 네 번째 해당하는 4를 서로 다른 것으로 간주하기 때문에 4444444인 28이 도출된다. 배열의 크기 N, 숫자가 더해지는 횟수 M, 그리고 K가 주어질 때 동빈이의 큰 수의 법칙에 따른 결과를 출력하시오

입력조건 : 첫째 줄에 N(2<= N<= 1000), M(1<=M<=10000), K(1<=K<=10000)의 자연수가 주어지며, 각 자연수는 공백으로 구분한다.

둘째 줄에 N개의 자연수가 주어진다. 각 자연수는 공백으로 구분한다. 단, 각각의 자연수는 1 이상 10000이하의 수로 주어진다

입력으로 주어지는 K는 항상 M보다 작거나 같다 첫째 줄에 동빈이의 큰 수의 법칙에 따라 더해진 답을 출력한다.

```
n, m, k = map(int, input().split()) # map은 int옵션을 통해 정수로 바꿀 수 있음
# input은 문자열로 입력받고, split()은 공백 기준으로 나눔
data= list(map(int,input().split())) # n개의 숫자 배열 입력
# 2 4 5를 입력하면 data = [2, 4, 5]로 저장됨

```

```
data.sort(reverse=True) # 내림차순 정렬
first=data[0] # 가장 큰 수
second=data[1] # 두 번째 큰 수

result=0

# 가장 큰수 최대한 더한 다음, 나머지를 그 다음 작은 수로 채우는 방법

while True:
  for i in range(k): #가장 큰 수를 k번 더하기
    if m==0: # m이 0이라면 반복문 탈출
      break
    result+=first
    m -= 1 # 더할 때마다 1씩 빼기
  if m==0: # m이 0이라면 반복문 탈출
    break
  result += second #두 번째로 큰 수를 한번 더하기
  m-=1 # 더할 때마다 1씩 빼기

print(result)
```

```
3 5 2
2 5 6
29
```

```
n, m, k = map(int, input().split())
data=list(map(int, input().split()))

data.sort()
first= data[n-1]
second=data[n-2]

# 가장 큰 수가 더해지는 횟수 계산
# k번 가장 큰 수+ 1번 두번째 큰 수 더하는 것이 1 세트이기 때문에 k+1을 나눠줌
count = (m//(k+1))*k
count += m % (k+1) # 남은 횟수는 모두 가장 큰 수로 더함

result=0
result += count * first
result += (m-count)*second
print(result)

```

```
3 5 2
2 5 6
29
```

3-3) 숫자 카드 게임

숫자 카드 게임은 여러 개의 숫자 카드 중에서 가장 높은 숫자가 쓰인 카드 한 장을 뽑는 게임이다. 단 게임의 룰은 다음과 같다.

1. 숫자가 쓰인 카드들이 n x m 형태로 놓여 있다. 이 때 n은 행의 개수이고 m은 열의 개수이다.
2. 먼저 뽑고자 하는 카드가 포함되어 있는 행을 선택한다.
3. 그 다음 선택된 행에 포함된 카드 중 가장 숫자가 낮은 카드를 뽑아야 한다.
4. 따라서 처음에 카드를 골라낼 행을 선택할 때, 이후에 해당 행에서 가장 숫자가 낮은 카드를 뽑을 것을 고려하여 최종적으로 가장 높은 숫자의 카드를 뽑을 수 있도록 전략을 세워야 한다.

입력조건

- 첫째 줄에 숫자카드들이 놓인 행의 개수 n과 열의 개수 m이 공백을 기준으로 하여 각각 자연수로 주어진다. (1<=n, m<=100)
- 둘째 줄부터 n개의 줄에 걸쳐 각 카드가 적힌 숫자가 주어진다. 각 숫자는 1 이상 10000 이하의 자연수이다.

출력조건

- 첫번째 게임의 룰에 맞게 선택한 카드에 적힌 숫자를 출력한다

```
n, m = map(int, input().split())

```

```
result = 0
for i in range(n):
  data=list(map(int, input().split()))
  min_value=min(data)
  result=max(result,min_value)

print(result)
```

```
3 3
4
7 2 7
1 7 4
4
```

```
n, m = map(int, input().split())

result=0
for i in range(n):
  data= list(map(int, input().split()))
  min_value=10001
  for a in data:
    min_value=min(min_value,a)
  result=max(result,min_value)

print(result)
```

```
3 3
5 7 3
3 8 5
6
6
```

3-4) 1이 될 때 까지

어떠한 수 n이 1이 될 때까지 다음의 두 과정 중 하나를 반복적으로 선택하여 수행하려고 한다. 단, 두 번째 연산은 n 이 k로 나누어떨어질 때만 선택할 수 있다.

1. n에서 1을 뺀다.
2. n을 k로 나눈다.

예를 들어 n이 17, k가 4라고 가정하자. 이 때 1번의 과정을 한 번 수행하면 n은 16이 된다.

이후에 2번의 과정을 두 번 수행하면 n은 1이 된다. 결과적으로 이 경우 전체 과정을 실행한 횟수는 3이 된다.

이는 n을 1으로 만드는 최소 횟수이다. n과 k가 주어질 때 n이 1이 될 때까지 1번 혹은 2번의 과정을 수행해야 하는 최소 횟수를 구하는 프로그램을 작성하시오.

입력조건

- 첫째 줄에 n(2<= n<=100000)과 k(2<= k <= 100000)가 공백으로 구분되며 각각 자연수로 주어진다. 이 때 입력으로 주어지는 n은 항상 k보다 크거나 같다.

출력조건

- 첫째 줄에 n이 1이 될 때까지 1번 혹은 2번의 과정을 수행해야 하는 횟수의 최솟값을 출력한다.

```
 n, k=map(int,input().split())

result=0
while True:
  if n==1:
    break

  if n%k==0:
    n= n//k
    result+=1
  else:
    n= n-1
    result+=1
```

```
14
15   print(result)
```

```
17 6
7
```

```
1 n, k = map(int, input().split())
2
3 result = 0
4
5 while n > 1:
6     if n % k == 0:
7         n //= k        # 정수 나눗셈
8     else:
9         n -= 1         # 1 빼기
10    result += 1        # 연산 횟수 누적
11
12 print(result)
13
```

```
17 6
7
```

```
1 n, k= map(int, input().split())
2 result=0
3
4 while n>=k:
5   while n%k != 0:
6     n -= 1
7     result += 1
8   n//=k
9   result+=1
10
11 print(result)
12
```

```
17 6
6
```

```
1 n,k= map(int, input().split())
2 result=0
3
4 while True:
5   target=(n//k)*k
6   result += (n-target)
7   n=target
8
9   if n<k:
10    break
11  result += 1
12  n //= k
13
14 result +=(n-1)
15 print(result)
```

```
17 6
7
```