[Tham Lam]. Bài 1. Đổi tiền

Tại ngân hàng có các mệnh giá bằng 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000. Tổng số tiền cần đổi có giá trị bằng N. Hãy xác định xem có ít nhất bao nhiêu tờ tiền sau khi đổi tiền?

Input Format

Dòng duy nhất chứa số nguyên N

Constraints

1<=N<=10^9

Output Format

In ra số tờ tiền tối thiểu

Sample Input 0

```
138
```

Sample Output 0

```
6
```

[Tham Lam]. Bài 2. Nhầm chữ số

Trong một buổi học toán, giáo viên viết 2 số nguyên, A và B, và yêu cầu Tèo thực hiện phép cộng. Tèo không bao giờ tính toán sai, nhưng thỉnh thoảng cậu ta chép các con số một cách không chính xác. Lỗi duy nhất của là ghi nhầm '5' thành '6' hoặc ngược lại. Cho hai số, A và B, tính tổng nhỏ nhất và lớn nhất mà Tèo có thể nhận được.

Input Format

1 dòng duy nhất chứa 2 số A và B

Constraints

1<=A<=B<=10^16

Output Format

In ra tổng lớn nhất và nhỏ nhất trên 1 dòng

Sample Input 0

```
891 746
```

Sample Output 0

```
1637 1636
```

[Tham Lam]. Bài 3. Max product sum

Cho mảng A[] gồm N phần tử, nhiệm vụ của bạn là sắp đặt lại vị trí các phần tử trong mảng và tính toán giá trị lớn nhất của biểu thức :

$$max = \sum_{i=0}^{n-1} A_i * i$$

Input Format

Dòng 1 chứa số nguyên dương N; Dòng 2 chứa N số nguyên của mảng A[] viết cách nhau một dấu cách

Constraints

1<=N<=10^6; 1<=A[i]<=10^9;

Output Format

In ra kết quả của bài toán chia dư với 10^9 + 7

Sample Input 0

```
6
8 1 7 9 8 1
```

Sample Output 0

```
116
```

[Tham Lam]. Bài 4. Chia tập

Cho mảng A[] gồm N số nguyên không âm và số K. Nhiệm vụ của bạn là hãy chia mảng A[] thành hai mảng con có kích cỡ K và N-K sao cho hiệu giữa tổng hai mảng con là lớn nhất. Ví dụ với mảng A[] = {8, 4, 5, 2, 10}, K=2 ta có kết quả là 17 vì mảng A[] được chia thành hai mảng {4, 2} và { 8, 5,10} có hiệu của hai mảng con là 23-6=17 là lớn nhất.

Input Format

Dòng duy nhất chứa 2 số nguyên N và K; Dòng thứ 2 gồm N số của mảng A[] **Constraints**

1<=K<=N<=10^6; 0<=A[i]<=10^9;

Output Format

In ra đáp án của bài toán

Sample Input 0

```
6 4
3 10 10 7 5 2
```

Sample Output 0

```
27
```

[Tham Lam]. Bài 5. Sắp xếp tham lam

Cho mảng A[] gồm N số và thực hiện các thao tác theo nguyên tắc dưới đây: 1. Ta chọn một mảng con sao cho phần tử ở giữa của mảng con cũng là phần tử ở giữa của mảng A[] (trong trường hợp N lẻ). 2. Đảo ngược mảng con đã chọn trong mảng A[]. Ta được phép chọn mảng con và phép đảo ngược mảng con bao nhiêu lần tùy ý. Ví dụ với mảng A[] = {1, 6, 3, 4, 5, 2, 7} ta có câu trả lời là YES vì: ta chọn mảng con {3, 4, 5} và đảo ngược để nhận được mảng A[]={1, 6, 5, 4, 3, 2, 7}, chọn tiếp mảng con {6, 5, 4, 3, 2} và đảo ngược ta nhận được mảng A[]={1, 2, 3, 4, 5, 6, 7}. Hãy cho biết ta có thể sắp xếp được mảng A[] bằng cách thực hiện các thao tác kể trên hay không?

Input Format

Dòng 1 chứa số nguyên dương N là số lẻ.
Dòng 2 chứa N số nguyên của mảng A[]

Constraints

1<=N<=10^6; 0<=A[i]<=10^9;

Output Format

In ra YES hoặc NO là đáp án của bài toán

Sample Input 0

```
5
1 3 8 7 3
```

Sample Output 0

```
NO
```

Sample Input 1

```
5
1 4 3 2 5
```

Sample Output 1

```
YES
```

[Tham Lam]. Bài 6. Max product of two array

Cho mảng A[], B[] đều có N phần tử. Nhiệm vụ của bạn là tìm giá trị lớn nhất của biểu thức P = A[0]*B[0] + A[1]*B[1] + ..+A[N-1]*B[N-1] bằng cách tráo đổi vị trí các phần tử của cả mảng A[] và B[].

Input Format

Dòng 1 chứa số nguyên dương N; Dòng 2 chứa N số nguyên của mảng A[]; Dòng
3 chứa N số nguyên của mảng B[];

Constraints

1<=N<=10^5; 0<=A[i], B[i]<=10^6

Output Format

In ra đáp án của bài toán

Sample Input 0

```
7
9 4 5 3 9 4 10
9 5 3 1 10 1 5
```

Sample Output 0

```
270
```

[Tham Lam]. Bài 7. Job Scheduling

Cho hệ gồm N hành động. Mỗi hành động được biểu diễn như một bộ đôi tương ứng với thời gian bắt đầu và thời gian kết thúc của mỗi hành động. Hãy tìm phương án thực hiện nhiều nhất các hành động nhất có thể, biết rằng 2 hành động phải được thực hiện một cách độc lập. 2 hành động được gọi là độc lập nếu thời gian kết thúc của hành động thứ nhất nhỏ hơn thời gian bắt đầu của hành động thứ 2.

Input Format

Dòng đầu tiên là số nguyên dương N; N dòng tiếp theo chứa thời gian bắt đầu và kết thúc của N hành động;

Constraints

1<=N<=10^6; 1<=Start[i]<=End[i]<=10^7;

Output Format

In ra số lượng hành động nhiều nhất có thể thực hiện.

Sample Input 0

```
16
1 5
2 7
3 7
5 7
6 7
10 12
10 13
1 3
7 8
9 14
5 6
9 10
3 5
8 13
1 6
3 6
```

Sample Output 0

```
4
```

[Tham Lam]. Bài 8. Job Scheduling with Profit

Cho N công việc. Mỗi công việc được biểu diễn như một bộ 3 số nguyên dương , trong đó JobId là mã của việc, Deadline là thời gian kết thúc của việc, Profit là lợi nhuận đem lại nếu hoàn thành việc đó đúng thời gian. Thời gian để hoàn toàn mỗi công việc là 1 đơn vị thời gian. Hãy cho biết lợi nhuận lớn nhất có thể thực hiện các việc với giả thiết mỗi việc được thực hiện đơn lẻ.

Gợi ý : B1. Sắp xếp công việc theo lợi nhuận giảm dần

B2. Đối với mỗi công việc => Lựa chọn mốc thời gian bắt đầu để thực hiện công việc, ví dụ nếu deadline là X thì mốc thời gian hợp lệ sẽ là 0,1...X-1, và lựa chọn mốc thời gian lớn nhất còn trống để thực hiện công việc này, sau đó đánh dấu mốc thời gian này đã được sử dụng.

```
struct job{
    int id, deadline, profit; };
```

Input Format

Dòng thứ 1 chứa số nguyên dương N; N dòng tiếp theo mô tả id, deadline, profit của N công việc

Constraints

1<=N<=10^5; 1<=JobID<=N; 1<=Deadline<=N; 1<=Profit<=1000;

Output Format

In ra lợi nhuận lớn nhất

Sample Input 0

```
4
1 4 20
2 1 10
3 1 40
4 1 30
```

Sample Output 0

```
60
```

[Tham Lam]. Bài 9. Nối dây 1

Cho N sợi dây, biết chi phí nối 2 sợ dây là tổng độ dài của 2 sợi dây đó. Nhiệm vụ của bạn là nối N sợi dây này thành 1 sao cho chi phí nối dây là nhỏ nhất

Input Format

Dòng 1 chứa số nguyên N; Dòng 2 chứa N số nguyên là độ dài các sợ dây

Constraints

1<=N<=10^5; Các sợi dây có độ dài không quá 10^5;

Output Format

In ra chi phí nối dây tối thiểu

Sample Input 0

```
7
7 7 6 10 4 8 3
```

Sample Output 0

```
124
```

Sample Input 1

```
4
4 3 2 6
```

Sample Output 1

```
29
```

[Tham Lam]. Bài 10. Nối dây 2

Cho N sợi dây, biết chi phí nối 2 sợ dây là tổng độ dài của 2 sợi dây đó. Nhiệm vụ của bạn là nối N sợi dây này thành 1 sao cho chi phí nối dây là lớn nhất

Input Format

Dòng 1 chứa số nguyên N; Dòng 2 chứa N số nguyên là độ dài các sợ dây

Constraints

1<=N<=10^5; Các sợi dây có độ dài không quá 10^9;

Output Format

Đáp án của bài toán chia dư với 10^9 + 7

Sample Input 0

```
9
10 1 5 7 4 8 7 7 1
```

Sample Output 0

```
305
```

Sample Input 1

```
3
5 6 1
```

Sample Output 1

```
23
```

[Tham lam]. Bài 11. Giá trị xâu kí tự

Cho xâu ký tự S[] bao gồm các ký tự in hoa [A, B, …,Z]. Ta định nghĩa giá trị của xâu S[] là tổng bình phương số lần xuất hiện mỗi ký tự trong xâu. Ví dụ với xâu S[] = “AAABBCD” ta có F(S) = 3^2 + 2^2 + 1^2 + 1^2 = 15. Hãy tìm giá trị nhỏ nhất của xâu S[] sau khi loại bỏ K ký tự trong xâu.

Input Format

Dòng đầu tiên đưa vào số lượng test T. Mỗi test được tổ chức thành 2 dòng. Dòng thứ nhất ghi lại số K. Dòng thứ 2 ghi lại xâu ký tự S[] có độ dài không vượt quá

10^6.

Constraints

T≤100; 1<=K<=10^6; 1<=len(S)<=10^6; Xâu S chỉ bao gồm các kí tự in hoa hoặc in thường.

Output Format

Đưa ra giá trị nhỏ nhất của mỗi test theo từng dòng.

Sample Input 0

```
2
0
ABCC
1
ABCC
```

Sample Output 0

```
6
3
```

[Tham lam]. Bài 12. Nối dây 3.

Cho N sợi dây với độ dài khác nhau được lưu trong mảng A[]. Nhiệm vụ của bạn là nối N sợi dây thành một sợi sao cho tổng chi phí nối dây là nhỏ nhất. Biết chi phí nối sợi dây thứ i và sợi dây thứ j là tổng độ dài hai sợi dây A[i] và A[j] Gợi ý : Sử dụng hàng đợi ưu tiên, priority_queue Tutorial : https://www.youtube.com/watch? v=DRcAJNhtwbY&t=559s&ab_channel=andrew2804

Input Format

Dòng thứ nhất đưa vào số lượng sợi dây N; Dòng tiếp theo đưa vào N số A[i] là độ dài của các sợi dây; Các số được viết cách nhau một vài khoảng trống.

Constraints

1≤N≤10^6; 1≤A[i]≤10^9.

Output Format

In ra chi phí nhỏ nhất lấy dư với (10^9 + 7).

Sample Input 0

```
5
4 2 7 6 9
```

Sample Output 0

```
62
```

[Tham Lam]. Bài 13. Sắp đặt xâu kí tự

Cho xâu ký tự S. Ta gọi giá trị của xâu S là tổng bình phương số lần xuất hiện mỗi ký tự trong S. Hãy tìm giá trị nhỏ nhất của xâu S sau khi thực hiện K lần loại bỏ ký tự.

Input Format

Dòng 1 chứa số nguyên K; Dòng 2 chứa xâu S;

Constraints

1<=K<=100000; 1<=len(S)<=100000;

Output Format

In ra đáp án của bài toán

Sample Input 0

```
2
adrwda
```

Sample Output 0

```
4
```

[Tham Lam]. Bài 14. Số may mắn

Hoàng yêu thích các số may mắn. Ta biết rằng một số là số may mắn nếu biểu diễn thập phân của nó chỉ chứa các chữ số may mắn là 4 và 7. Ví dụ, các số 47, 744, 4 là số may mắn và 5, 17, 467 không phải. Hoàng muốn tìm số may mắn bé nhất có tổng các chữ số bằng n. Hãy giúp anh ấy

Input Format

Dòng duy nhất chứa số nguyên dương n

Constraints

1<=n<=10^6;

Output Format

In ra đáp án của bài toán, nếu không tồn tại đáp án thì in ra -1

Sample Input 0

```
16
```

Sample Output 0

```
4444
```

Sample Input 1

```
14
```

Sample Output 1

```
77
```

[Tham Lam]. Bài 15. Sắp đặt xâu kí tự

Cho xâu kí tự S chỉ bao gồm các kí tự in thường, hãy kiểm tra xem có thể sắp đặt lại các kí tự trong xâu sao cho không có 2 kí tự kề nhau nào giống nhau hay không?

Input Format

Dòng duy nhất chứa xâu S

Constraints

1<=len(S)<=10000;

Output Format

Nếu có thể sắp đặt lại xâu kí tự in ra YES, ngược lại in ra NO.

Sample Input 0

```
aqeaaqwq
```

Sample Output 0

```
YES
```

[Tham Lam]. Bài 16. Mua lương thực SPOJ

Giả sử bạn là một người nghèo trong địa phương của bạn. Địa phương của bạn có duy nhất một cửa hàng bán lương thực. Cửa hàng của bạn mở cửa tất cả các ngày trong tuần ngoại trừ chủ nhật. Cho bộ ba số N, S, M thỏa mãn ràng buộc sau: N : số đơn vị lương thực nhiều nhất bạn có thể mua trong ngày. S : số lượng ngày bạn cần được sử dụng lương thực để tồn tại. M : số đơn vị lương thực cần có mỗi ngày để bạn tồn tại. Giả sử bạn đang ở ngày thứ 2 trong tuần và cần tồn tại trong S ngày tới. Hãy cho biết số lượng ngày ít nhất bạn cần phải mua lương thực từ của hàng để tồn tại hoặc bạn sẽ bị chết đói trong S ngày tới.

Input Format

1 dòng chứa 3 số N, S, M

Constraints

1<=S,N,M<=100

Output Format

In ra số ngày ít nhất cần mua lương thực, nếu không thể mua đủ lương thực để tồn tại thì in ra -1

Sample Input 0

```
7 5 7
```

Sample Output 0

```
5
```

[Tham Lam]. Bài 17. Số nhỏ nhất

Cho hai số nguyên dương S và D, trong đó S là tổng các chữ số và D là số các chữ số của một số. Nhiệm vụ của bạn là tìm số nhỏ nhất thỏa mãn S và D?

Input Format

1 dòng gồm 2 số S, D

Constraints

1<=S,D<=1000;

Output Format

In ra số nhỏ nhất có D chữ số và có tổng bằng S, nếu không tìm được đáp án thì in ra -1

Sample Input 0

```
12 8
```

Sample Output 0

```
10000029
```

[Tham Lam]. Bài 18. Phân số đơn vị (egyptian fractions)

Một phân số đơn vị nếu tử số của phân số đó là 1. Mọi phân số nguyên dương đều có thể biểu diễn thành tổng các phân số đơn vị. Ví dụ 2/3 = 1/2 + 1/6. Cho phân số nguyên dương P/Q bất kỳ , hãy biểu diễn phân số nguyên dương thành tổng phân số đơn vị với số hạng tử là ít nhất.

Input Format

1 dòng duy nhất chứa 2 số P, Q

Constraints

1<=P,Q<=200

Output Format

Đưa ra đáp án trên 1 dòng

Sample Input 0

```
9 6
```

Sample Output 0

```
1/1 + 1/2
```

Sample Input 1

```
5 6
```

Sample Output 1

```
1/2 + 1/3
```

[Tham Lam]. Bài 19. Tích lớn nhất

Cho dãy số A gồm N phần tử là các số nguyên. Hãy tính tích lớn nhất của 2 hoặc 3 phần tử trong dãy.

Input Format

Dòng đầu tiên là N; Dòng thứ 2 là N phần tử của mảng A

Constraints

1<=N<=1000; 0<=abs(A[i])<=10^6

Output Format

In ra tích lớn nhất của 2 hoặc 3 phần tử trong mảng

Sample Input 0

```
5
-9 4 3 -3 1
```

Sample Output 0

```
108
```

[Tham Lam]. Bài 20. Taxi

Có N nhóm học sinh, mỗi nhóm học sinh có từ 1 tới 4 người. Các nhóm học sinh này dự định sẽ đi thăm quan vườn bách thú bằng những chiếc xe taxi, mỗi xe taxi trở được tối đa 4 người. Hãy tìm số lượng taxi tối thiểu để có thể trở hết N nhóm học sinh này, biết rằng những học sinh ở cùng 1 nhóm sẽ đi cùng 1 taxi

Input Format

Dòng đầu tiên chứa số nguyên dương N là số nhóm học sinh; Dòng thứ 2 gồm N số là số lượng của các nhóm học sinh

Constraints

1<=N<=10000; Số lượng học sinh của mỗi nhóm là 1 số dương không quá 4

Output Format

In ra số lượng xe taxi tối thiểu cần dùng

Sample Input 0

```
6
2 1 3 1 2 2
```

Sample Output 0

```
3
```

[Tham Lam]. Bài 21. Di chuyển dấu ngoặc

Cho một xâu kí tự S chỉ bao gồm các kí tự '(' hoặc kí tự ')'. S có độ dài là số chẵn có giá trị N. Xâu S gồm N / 2 kí tự mở ngoặc và N / 2 kí tự đóng ngoặc. Ở mỗi thao tác các bạn được lựa chọn 1 kí tự bất kì của S để đưa về vị trí đầu tiên hoặc vị trí cuối cùng của dãy. Các bạn hãy xác định số thao tác tối thiểu cần thực hiện để tạo được 1 xâu dấu ngoặc hợp lệ. Một số ví dụ về xâu hợp lệ : (), ((())), ()((()))...

Input Format

Một dòng duy nhất chứa xâu S

Constraints

1<=N<=1000;

Output Format

In ra số thao tác tối thiểu cần thực hiện

Sample Input 0

```
)))(((()))
```

Sample Output 0

```
3
```

[Tham Lam]. Bài 22. Cặp số

Ta gọi một cặp số (x, y) là tương tự nhau nếu chúng có cùng tính chất chẵn lẻ hoặc có abs(x - y) = 1. Bạn được cung cấp một mảng A[] có N phần tử, hãy kiểm tra xem có thể chia N phần tử này thành các cặp, sao cho mỗi cặp số đều tương tự nhau.

Input Format

Dòng đầu tiên chứa số nguyên dương N là số chẵn. Dòng 2 chứa N số nguyên của mảng A[]

Constraints

1<=N<=100; 1<=A[i]<=1000;

Output Format

In ra YES nếu có thể chia thành các cặp tương tự, ngược lại in ra NO

Sample Input 0

```
6
78 17 17 42 11 43
```

Sample Output 0

```
YES
```

[Tham Lam]. Bài 23. Tích của 3 số

Cho số nguyên dương N, nhiệm vụ của bạn là kiểm tra xem có thể viết N = a * b * c hay không, trong đó a, b, c là 3 số nguyên dương lớn hơn hoặc bằng 2 khác nhau.

Input Format

Dòng duy nhất chứa số nguyên dương N

Constraints

2<=N<=10^9

Output Format

In ra YES nếu có thể biểu diễn N dưới dạng tích của 3 số, ngược lại in ra NO **Sample Input 0**

```
11
```

Sample Output 0

```
NO
```

Sample Input 1

```
24
```

Sample Output 1

```
YES
```

[Tham Lam]. Bài 24. Xóa xâu kí tự

Tí vào Tèo cùng chơi một trò chơi với xâu nhị phân S. Xâu nhị phân S chỉ bao gồm các kí tự 0 và 1. Ở mỗi bước đi, 2 bạn nhỏ có thể chọn 1 xâu con liên tiếp các kí tự giống nhau ở xâu S và xóa nó khỏi xâu S. Sau khi xóa 1 xâu thì 2 xâu bên trái và phải của xâu vừa xóa sẽ trở thành liền kề. Ban đầu Tí là người đi trước, sau đó 2 bạn lần lượt thực hiện bước đi của mình cho tới khi xâu S trở thành xâu rỗng. Bạn hãy xác định xem Tí có thể xóa tối đa bao nhiêu kí tự 1 biết rằng cả 2 bạn đều chơi tối ưu

Input Format

Dòng duy nhất chứa xâu S

Constraints

1<=len(S)<=100000;

Output Format

In ra số lượng số 1 tối đa mà Tí có thể xóa được

Sample Input 0

```
1000101110011111
```

Sample Output 0

```
6
```

[Tham Lam]. Bài 25. String game

Tí vào Tèo chơi một trò chơi với xâu kí tự. Luật chơi như sau, ở mỗi lượt chơi 2 người có thể lựa chọn 1 trong 2 thao tác : 1. Hai người đi theo lượt, Tí là người đi trước, ở mỗi lượt đi 2 bạn nhỏ có thể lựa chọn 1 kí tự bất kỳ và xóa kí tự này khỏi xâu S. 2.Lấy xâu S sau đó sắp đặt lại các kí tự trong xâu sao cho nó là một xâu đối xứng thì người đó sẽ thắng. Biết rằng 2 bạn đều chơi tối ưu, bạn hãy xác định xem ai là người chiến thắng ?

Input Format

Dòng duy nhất chứa xâu S

Constraints

S chỉ bao gồm các kí tự in thường và có độ dài không quá 10000

Output Format

Nếu Tí thắng in ra 1, ngược lại nếu Tèo thắng in ra 2

Sample Input 0

```
kpjdpgb
```

Sample Output 0

```
1
```

Sample Input 1

```
safjaqih
```

Sample Output 1

```
2
```

[Tham Lam]. Bài 26. Hàng đợi

Cô bé Anna đi mua sắm cùng mẹ và cô băn khoăn không biết làm thế nào để cải thiện chất lượng dịch vụ.

Có n người trong hàng đợi. Đối với mỗi người, chúng tôi biết thời gian cần thiết t để phục vụ anh ta. Một người sẽ thất vọng nếu thời gian anh ta chờ đợi nhiều hơn thời gian cần thiết để phục vụ anh ta. Thời gian một người chờ là tổng thời gian tất cả những người đứng trong hàng đợi trước mặt anh ta được phục vụ. Anna nghĩ rằng nếu chúng ta hoán đổi một số người trong hàng đợi, thì chúng ta có thể giảm số người thất vọng.

Giúp Anna tìm ra con số tối đa mà những người không thất vọng có thể đạt được bằng cách hoán đổi những người trong hàng đợi.

Input Format

Dòng đầu tiên chứa số N là số người trong hàng đợi; Dòng thứ 2 chứa N số là thời gian cần phục vụ của N người

Constraints

1<=N<=10^5; 1<=t<=10^9;

Output Format

In ra đáp án của bài toán

Sample Input 0

```
7
4 3 17 4 5 14 20
```

Sample Output 0

```
3
```

Minimum Absolute Difference in an Array

The absolute difference is the positive difference between two values a and b, is written $|a - b|$ or $|b - a|$ and they are equal. If $a = 3$ and $b = 2$, $|3 - 2| = |2 - 3| = 1$. Given an array of integers, find the minimum absolute difference between any two elements in the array.

Example. $arr = [-2, 2, 4]$

There are 3 pairs of numbers: $[-2, 2], [-2, 4]$ and $[2, 4]$. The absolute differences for these pairs are $|(-2) - 2| = 4$, $|(-2) - 4| = 6$ and $|2 - 4| = 2$. The minimum absolute difference is 2.

Function Description

Complete the *minimumAbsoluteDifference* function in the editor below. It should return an integer that represents the minimum absolute difference between any pair of elements.

minimumAbsoluteDifference has the following parameter(s):

- *int arr[n]:* an array of integers

Returns

- *int:* the minimum absolute difference found

Input Format

The first line contains a single integer n, the size of arr.
The second line contains n space-separated integers, $arr[i]$.

Constraints

- $2 \leq n \leq 10^5$
- $-10^9 \leq arr[i] \leq 10^9$

Sample Input 0

```
3
3 -7 0
```

Sample Output 0

```
3
```

Explanation 0

The first line of input is the number of array elements. The array, $arr = [3, -7, 0]$ There are three pairs to test: $(3, -7)$, $(3, 0)$, and $(-7, 0)$. The absolute differences are:

- $|3 - -7| \Rightarrow 10$
- $|3 - 0| \Rightarrow 3$
- $|-7 - 0| \Rightarrow 7$

Remember that the order of values in the subtraction does not influence the result. The smallest of these absolute differences is 3.

Sample Input 1

```
10
-59 -36 -13 1 -53 -92 -2 -96 -54 75
```

Sample Output 1

```
1
```

Explanation 1

The smallest absolute difference is $|-54 - -53| = 1$.

Sample Input 2

```
5
1 -3 71 68 17
```

Sample Output 2

```
3
```

Explanation 2

The minimum absolute difference is $|71 - 68| = 3$.

Marc's Cakewalk

Marc loves cupcakes, but he also likes to stay fit. Each cupcake has a calorie count, and Marc can walk a distance to expend those calories. If Marc has eaten j cupcakes so far, after eating a cupcake with c calories he must walk *at least* $2^j \times c$ miles to maintain his weight.

Example
$calorie = [5, 10, 7]$

If he eats the cupcakes in the order shown, the miles he will need to walk are
$(2^0 \times 5) + (2^1 \times 10) + (2^2 \times 7) = 5 + 20 + 28 = 53$. This is not the minimum, though, so we need to test other orders of consumption. In this case, our minimum miles is calculated as
$(2^0 \times 10) + (2^1 \times 7) + (2^2 \times 5) = 10 + 14 + 20 = 44$.

Given the individual calorie counts for each of the cupcakes, determine the minimum number of miles Marc must walk to maintain his weight. Note that he can eat the cupcakes *in any order*.

Function Description

Complete the *marcsCakewalk* function in the editor below.

marcsCakewalk has the following parameter(s):

- *int calorie[n]:* the calorie counts for each cupcake

Returns

- *long:* the minimum miles necessary

Input Format

The first line contains an integer n, the number of cupcakes in $calorie$.
The second line contains n space-separated integers, $calorie[i]$.

Constraints

- $1 \leq n \leq 40$
- $1 \leq c[i] \leq 1000$

Sample Input 0

```
3
1 3 2
```

Sample Output 0

```
11
```

Explanation 0

Let's say the number of miles Marc must walk to maintain his weight is $miles$. He can minimize $miles$ by eating the $n = 3$ cupcakes in the following order:

1. Eat the cupcake with $c_1 = 3$ calories, so $miles = 0 + (3 \cdot 2^0) = 3$.
2. Eat the cupcake with $c_2 = 2$ calories, so $miles = 3 + (2 \cdot 2^1) = 7$.
3. Eat the cupcake with $c_0 = 1$ calories, so $miles = 7 + (1 \cdot 2^2) = 11$.

We then print the final value of $miles$, which is 11, as our answer.

Sample Input 1

```
4
7 4 9 6
```

Sample Output 1

```
79
```

Explanation 1

$(2^0 * 9) + (2^1 * 7) + (2^2 * 6) + (2^3 * 4) = 9 + 14 + 24 + 32 = 79$

Grid Challenge

Given a square grid of characters in the range ascii[a-z], rearrange elements of each row alphabetically, ascending. Determine if the columns are also in ascending alphabetical order, top to bottom. Return `YES` if they are or `NO` if they are not.

Example
$grid = [\text{'abc'},\text{'ade'},\text{'efg'}]$

The grid is illustrated below.

```
a b c
a d e
e f g
```

The rows are already in alphabetical order. The columns `a a e`, `b d f` and `c e g` are also in alphabetical order, so the answer would be `YES`. Only elements within the same row can be rearranged. They cannot be moved to a different row.

Function Description

Complete the *gridChallenge* function in the editor below.

gridChallenge has the following parameter(s):

- *string grid[n]:* an array of strings

Returns

- *string:* either `YES` or `NO`

Input Format

The first line contains t, the number of testcases.

Each of the next t sets of lines are described as follows:
- The first line contains n, the number of rows and columns in the grid.
- The next n lines contains a string of length n

Constraints

$1 \leq t \leq 100$
$1 \leq n \leq 100$
Each string consists of lowercase letters in the range ascii[a-z]

Output Format

For each test case, on a separate line print `YES` if it is possible to rearrange the grid alphabetically ascending in both its rows and columns, or `NO` otherwise.

Sample Input

```
STDIN   Function
-----   --------
1       t = 1
5       n = 5
ebacd   grid = ['ebacd', 'fghij', 'olmkn', 'trpqs', 'xywuv']
fghij
olmkn
trpqs
xywuv
```

Sample Output

```
YES
```

Explanation

The 5x5 grid in the 1 test case can be reordered to

```
abcde
fghij
klmno
pqrst
uvwxy
```

This fulfills the condition since the rows 1, 2, ..., 5 and the columns 1, 2, ..., 5 are all alphabetically sorted.

Luck Balance

Lena is preparing for an important coding competition that is preceded by a number of sequential preliminary contests. Initially, her luck balance is 0. She believes in "saving luck", and wants to check her theory. Each contest is described by two integers, $L[i]$ and $T[i]$:

- $L[i]$ is the amount of luck associated with a contest. If Lena *wins* the contest, her luck balance will *decrease* by $L[i]$; if she *loses* it, her luck balance will *increase* by $L[i]$.
- $T[i]$ denotes the contest's *importance rating*. It's equal to 1 if the contest is *important*, and it's equal to 0 if it's *unimportant*.

If Lena loses no more than k *important* contests, what is the maximum amount of luck she can have after competing in all the preliminary contests? This value *may* be negative.

Example
$k = 2$
$L = [5, 1, 4]$
$T = [1, 1, 0]$

```
Contest         L[i]    T[i]
1               5       1
2               1       1
3               4       0
```

If Lena loses all of the contests, her will be $5 + 1 + 4 = 10$. Since she is allowed to lose 2 important contests, and there are only 2 important contests, she can lose all three contests to maximize her luck at 10.

If $k = 1$, she has to win at least 1 of the 2 important contests. She would choose to win the lowest value important contest worth 1. Her final luck will be $5 + 4 - 1 = 8$.

Function Description

Complete the *luckBalance* function in the editor below.

luckBalance has the following parameter(s):

- *int k*: the number of important contests Lena can lose
- *int contests[n][2]:* a 2D array of integers where each $contests[i]$ contains two integers that represent the luck balance and importance of the i^{th} contest

Returns

- *int:* the maximum luck balance achievable

Input Format

The first line contains two space-separated integers n and k, the number of preliminary contests and the maximum number of important contests Lena can lose.

Each of the next n lines contains two space-separated integers, $L[i]$ and $T[i]$, the contest's luck balance and its importance rating.

Constraints

- $1 \leq n \leq 100$
- $0 \leq k \leq N$
- $1 \leq L[i] \leq 10^4$
- $T[i] \in \{0, 1\}$

Sample Input

```
STDIN       Function
-----       --------
6 3         n = 6, k = 3
5 1         contests = [[5, 1], [2, 1], [1, 1], [8, 1], [10, 0], [5, 0]]
2 1
1 1
8 1
10 0
5 0
```

Sample Output

```
29
```

Explanation

There are $n = 6$ contests. Of these contests, 4 are important and she cannot lose more than $k = 3$ of them. Lena maximizes her luck if she wins the 3^{rd} important contest (where $L[i] = 1$) and loses all of the other five contests for a total luck balance of $5 + 2 + 8 + 10 + 5 - 1 = 29$.

Maximum Perimeter Triangle

Given an array of stick lengths, use 3 of them to construct a non-degenerate triangle with the maximum possible perimeter. Return an array of the lengths of its sides as 3 integers in non-decreasing order.

If there are several valid triangles having the maximum perimeter:

1. Choose the one with the *longest maximum side*.
2. If more than one has that maximum, choose from them the one with the *longest minimum side*.
3. If more than one has that maximum as well, print any one them.

If no non-degenerate triangle exists, return $[-1]$.

Example
$sticks = [1, 2, 3, 4, 5, 10]$

The triplet $(1, 2, 3)$ will not form a triangle. Neither will $(4, 5, 10)$ or $(2, 3, 5)$, so the problem is reduced to $(2, 3, 4)$ and $(3, 4, 5)$. The longer perimeter is $3 + 4 + 5 = 12$.

Function Description

Complete the *maximumPerimeterTriangle* function in the editor below.

maximumPerimeterTriangle has the following parameter(s):

- *int sticks[n]:* the lengths of sticks available

Returns

- *int[3] or int[1]:* the side lengths of the chosen triangle in non-decreasing order or -1

Input Format

The first line contains single integer n, the size of array $sticks$.
The second line contains n space-separated integers $sticks[i]$, each a stick length.

Constraints

- $3 \leq n \leq 50$
- $1 \leq sticks[i] \leq 10^9$

Sample Input 0

```
5
1 1 1 3 3
```

Sample Output 0

```
1 3 3
```

Explanation 0

There are 2 possible unique triangles:

1. $(1, 1, 1)$
2. $(1, 3, 3)$

The second triangle has the largest perimeter, so we print its side lengths on a new line in non-decreasing order.

Sample Input 1

```
3
1 2 3
```

Sample Output 1

```
-1
```

Explanation 1

The triangle $(1, 2, 3)$ is degenerate and thus can't be constructed, so we print `-1` on a new line.

Sample Input 2

```
6
1 1 1 2 3 5
```

Sample Output 2

```
1 1 1
```

Explanation 2

The triangle (1,1,1) is the only valid triangle.

Candies

HackerRank

Alice is a kindergarten teacher. She wants to give some candies to the children in her class. All the children sit in a line and each of them has a rating score according to his or her performance in the class. Alice wants to give at least 1 candy to each child. If two children sit next to each other, then the one with the higher rating must get more candies. Alice wants to minimize the total number of candies she must buy.

Example

$arr = [4, 6, 4, 5, 6, 2]$

She gives the students candy in the following minimal amounts: $[1, 2, 1, 2, 3, 1]$. She must buy a minimum of *10* candies.

Function Description

Complete the *candies* function in the editor below.

candies has the following parameter(s):

- *int n:* the number of children in the class
- *int arr[n]:* the ratings of each student

Returns

- *int:* the minimum number of candies Alice must buy

Input Format

The first line contains an integer, n, the size of arr.
Each of the next n lines contains an integer $arr[i]$ indicating the rating of the student at position i.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 10^5$

Sample Input 0

```
3
1
2
2
```

Sample Output 0

```
4
```

Explanation 0

Here 1, 2, 2 is the rating. Note that when two children have equal rating, they are allowed to have different number of candies. Hence optimal distribution will be 1, 2, 1.

Sample Input 1

```
10
2
4
2
6
1
7
8
9
2
1
```

Sample Output 1

```
19
```

Explanation 1

Optimal distribution will be $1, 2, 1, 2, 1, 2, 3, 4, 2, 1$

Sample Input 2

```
8
2
4
3
5
2
6
4
5
```

Sample Output 2

```
12
```

Explanation 2

Optimal distribution will be 12121212.

Sherlock and The Beast

Sherlock Holmes suspects his archenemy Professor Moriarty is once again plotting something diabolical. Sherlock's companion, Dr. Watson, suggests Moriarty may be responsible for MI6's recent issues with their supercomputer, *The Beast*.

Shortly after resolving to investigate, Sherlock receives a note from Moriarty boasting about infecting *The Beast* with a virus. He also gives him a clue: an integer. Sherlock determines the key to removing the virus is to find the largest *Decent Number* having that number of digits.

A *Decent Number* has the following properties:

1. Its digits can only be *3*'s and/or *5*'s.
2. The number of *3*'s it contains is divisible by *5*.
3. The number of *5*'s it contains is divisible by *3*.
4. It is the largest such number for its length.

Moriarty's virus shows a clock counting down to *The Beast*'s destruction, and time is running out fast. Your task is to help Sherlock find the key before *The Beast* is destroyed!

For example, the numbers 55533333 and 555555 are both decent numbers because there are 3 5's and 5 3's in the first, and 6 5's in the second. They are the largest values for those length numbers that have proper divisibility of digit occurrences.

Function Description

Complete the *decentNumber* function in the editor below.

decentNumber has the following parameter(s):

- *int n:* the length of the decent number to create

Prints

Print the decent number for the given length, or -1 if a decent number of that length cannot be formed. No return value is expected.

Input Format

The first line is an integer, t, the number of test cases.

The next t lines each contain an integer n, the number of digits in the number to create.

Constraints

$1 \leq t \leq 20$
$1 \leq n \leq 100000$

Sample Input

```
STDIN   Function
-----   --------
4       t = 4
1       n = 1 (first test case)
3       n = 3 (second test case)
5
11
```

Sample Output

```
-1
555
33333
55555533333
```

Explanation

For $n = 1$, there is no *Decent Number* having 1 digit, so print -1.
For $n = 3$, 555 is the only possible number. (*Decent Number Property 3*).
For $n = 5$, 33333 is the only possible number. (*Decent Number Property 2*).
For $n = 11$, 55555533333 is the *Decent Number*. All other permutations of these digits are not *decent* (*Decent Number Property 4*).

Priyanka and Toys

HackerRank

Priyanka works for an international toy company that ships by container. Her task is to the determine the lowest cost way to combine her orders for shipping. She has a list of item weights. The shipping company has a requirement that all items loaded in a container must weigh less than or equal to 4 units plus the weight of the minimum weight item. All items meeting that requirement will be shipped in one container.

What is the smallest number of containers that can be contracted to ship the items based on the given list of weights?

For example, there are items with weights $w = [1, 2, 3, 4, 5, 10, 11, 12, 13]$. This can be broken into two containers: $[1, 2, 3, 4, 5]$ and $[10, 11, 12, 13]$. Each container will contain items weighing within 4 units of the minimum weight item.

Function Description

Complete the *toys* function in the editor below. It should return the minimum number of containers required to ship.

toys has the following parameter(s):

- *w*: an array of integers that represent the weights of each order to ship

Input Format

The first line contains an integer n, the number of orders to ship.
The next line contains n space-separated integers, $w[1], w[2], \ldots, w[n]$, representing the orders in a weight array.

Constraints

$1 \leq n \leq 10^5$
$0 \leq w[i] \leq 10^4, where\ i \in [1, n]$

Output Format

Return the integer value of the number of containers Priyanka must contract to ship all of the toys.

Sample Input

```
8
1 2 3 21 7 12 14 21
```

Sample Output

```
4
```

Explanation

The first container holds items weighing 1, 2 and 3. (weights in range $1 \ldots 5$)
The second container holds the items weighing 21 units. ($21 \ldots 25$)
The third container holds the item weighing 7 units. ($7 \ldots 11$)
The fourth container holds the items weighing 12 and 14 units. ($12 \ldots 14$)

4 containers are required.

Largest Permutation

HackerRank

You are given an unordered array of unique integers incrementing from 1. You can swap any two elements a limited number of times. Determine the largest lexicographical value array that can be created by executing no more than the limited number of swaps.

Example
$arr = [1, 2, 3, 4]$
$k = 1$

The following arrays can be formed by swapping the 1 with the other elements:

```
[2,1,3,4]
[3,2,1,4]
[4,2,3,1]
```

The highest value of the four (including the original) is $[4, 2, 3, 1]$. If $k \geq 2$, we can swap to the highest possible value: $[4, 3, 2, 1]$.

Function Description

Complete the *largestPermutation* function in the editor below. It must return an array that represents the highest value permutation that can be formed.

largestPermutation has the following parameter(s):

- *int k:* the maximum number of swaps
- *int arr[n]:* an array of integers

Input Format

The first line contains two space-separated integers n and k, the length of arr and the maximum swaps that can be performed. The second line contains n distinct space-separated integers from 1 to n as $arr[i]$ where $1 \leq arr[i] \leq n$.

Constraints

$1 \leq n \leq 10^5$
$1 \leq k \leq 10^9$

Output Format

Print the lexicographically largest permutation you can make with **at most** k swaps.
Sample Input 0

```
STDIN       Function
-----       --------
5 1         n = 5, k = 1
4 2 3 5 1   arr = [4, 2, 3, 5, 1]
```

Sample Output 0

```
5 2 3 4 1
```

Explanation 0

You can swap any two numbers in $[4, 2, 3, 5, 1]$ and see the largest permutation is $[5, 2, 3, 4, 1]$

Sample Input 1

```
3 1
2 1 3
```

Sample Output 1

```
3 1 2
```

Explanation 1

With 1 swap we can get $[1, 2, 3]$, $[3, 1, 2]$ and $[2, 3, 1]$. Of these, $[3, 1, 2]$ is the largest permutation.

Sample Input 2

```
2 1
2 1
```

Sample Output 2

```
2 1
```

Explanation 2

We can see that $[2, 1]$ is already the largest permutation. We don't make any swaps.

Mark and Toys

HackerRank

Mark and Jane are very happy after having their first child. Their son loves toys, so Mark wants to buy some. There are a number of different toys lying in front of him, tagged with their prices. Mark has only a certain amount to spend, and he wants to maximize the number of toys he buys with this money. Given a list of toy prices and an amount to spend, determine the maximum number of gifts he can buy.

Note Each toy can be purchased only once.

Example
$prices = [1, 2, 3, 4]$
$k = 7$

The budget is 7 units of currency. He can buy items that cost $[1, 2, 3]$ for 6, or $[3, 4]$ for 7 units. The maximum is 3 items.

Function Description

Complete the function *maximumToys* in the editor below.

maximumToys has the following parameter(s):

- *int prices[n]:* the toy prices
- *int k:* Mark's budget

Returns

- *int:* the maximum number of toys

Input Format

The first line contains two integers, n and k, the number of priced toys and the amount Mark has to spend.
The next line contains n space-separated integers $prices[i]$

Constraints

$1 \leq n \leq 10^5$
$1 \leq k \leq 10^9$
$1 \leq prices[i] \leq 10^9$
A toy can't be bought multiple times.

Sample Input

```
7 50
1 12 5 111 200 1000 10
```

Sample Output

```
4
```

Explanation

He can buy only 4 toys at most. These toys have the following prices: $1, 12, 5, 10$.

Max Min

You will be given a list of integers, arr, and a single integer k. You must create an array of length k from elements of arr such that its *unfairness* is minimized. Call that array arr'. Unfairness of an array is calculated as

$$max(arr') - min(arr')$$

Where:
- *max* denotes the largest integer in arr'.
- *min* denotes the smallest integer in arr'.

Example

$arr = [1, 4, 7, 2]$
$k = 2$

Pick any two elements, say $arr' = [4, 7]$.
$unfairness = max(4, 7) - min(4, 7) = 7 - 4 = 3$
Testing for all pairs, the solution $[1, 2]$ provides the minimum unfairness.

Note: Integers in arr may not be unique.

Function Description

Complete the *maxMin* function in the editor below.
maxMin has the following parameter(s):

- *int k:* the number of elements to select
- *int arr[n]:*: an array of integers

Returns

- *int:* the minimum possible *unfairness*

Input Format

The first line contains an integer n, the number of elements in array arr.
The second line contains an integer k.
Each of the next n lines contains an integer $arr[i]$ where $0 \leq i < n$.

Constraints

$2 \leq n \leq 10^5$
$2 \leq k \leq n$
$0 \leq arr[i] \leq 10^9$

Sample Input 0

```
7
3
10
100
300
200
1000
20
30
```

Sample Output 0

```
20
```

Explanation 0

Here $k = 3$; selecting the 3 integers $10, 20, 30$, unfairness equals

```
max(10,20,30) - min(10,20,30) = 30 - 10 = 20
```

Sample Input 1

```
10
4
1
2
3
4
10
20
30
40
100
200
```

Sample Output 1

```
3
```

Explanation 1

Here $k = 4$; selecting the 4 integers $1, 2, 3, 4$, unfairness equals

```
max(1,2,3,4) - min(1,2,3,4) = 4 - 1 = 3
```

Sample Input 2

```
5
2
1
2
1
```

```
2
1
```

Sample Output 2

```
0
```

Explanation 2

Here $k = 2$. $arr' = [2, 2]$ or $arr' = [1, 1]$ give the minimum unfairness of 0.

Jim and the Orders

HackerRank

Jim's Burgers has a line of hungry customers. Orders vary in the time it takes to prepare them. Determine the order the customers receive their orders. Start by numbering each of the customers from 1 to n, front of the line to the back. You will then be given an *order number* and a *preparation time* for each customer.

The time of delivery is calculated as the sum of the order number and the preparation time. If two orders are delivered at the same time, assume they are delivered in ascending customer number order.

For example, there are $n = 5$ customers in line. They each receive an order number $order[i]$ and a preparation time $prep[i]$.:

```
Customer        1       2       3       4       5
Order #         8       5       6       2       4
Prep time       3       6       2       3       3
Calculate:
Serve time      11      11      8       5       7
```

We see that the orders are delivered to customers in the following order:

```
Order by:
Serve time      5       7       8       11      11
Customer        4       5       3       1       2
```

Function Description

Complete the *jimOrders* function in the editor below. It should return an array of integers that represent the order that customers' orders are delivered.

jimOrders has the following parameter(s):

- *orders*: a 2D integer array where each $orders[i]$ is in the form $[order[i], prep[i]]$.

Input Format

The first line contains an integer n, the number of customers.
Each of the next n lines contains two space-separated integers, an order number and prep time for $customer[i]$.

Constraints

- $1 \leq n \leq 10^3$
- $1 \leq i \leq n$
- $1 \leq order[i], prep[i] \leq 10^6$

Output Format

Print a single line of n space-separated customer numbers (recall that customers are numbered from 1 to n) that describes the sequence in which the customers receive their burgers. If two or more customers receive their burgers at the same time, print their numbers in ascending order.

Sample Input 0

```
3
1 3
2 3
3 3
```

Sample Output 0

```
1 2 3
```

Explanation 0

Jim has the following orders:

1. $order[1] = 1, prep[1] = 3$. This order is delivered at time $t = 1 + 3 = 4$.
2. $order[2] = 2, prep[2] = 3$. This order is delivered at time $t = 2 + 3 = 5$.
3. $order[3] = 3, prep[3] = 3$. This order is delivered at time $t = 3 + 3 = 6$.

The orders were delivered in the same order as the customers stood in line. The index in $order[i]$ is the customer number and is what is printed. In this case, the customer numbers match the order numbers.

Sample Input 1

```
5
8 1
4 2
5 6
3 1
4 3
```

Sample Output 1

```
4 2 5 1 3
```

Explanation 1

Jim has the following orders:

1. $order[1] = 8, prep[1] = 1$. This order is delivered at time $t = 8 + 1 = 9$.
2. $order[2] = 4, prep[2] = 2$. This order is delivered at time $t = 4 + 2 = 6$.
3. $order[3] = 5, prep[3] = 6$. This order is delivered at time $t = 5 + 6 = 11$.
4. $order[4] = 3, prep[4] = 1$. This order is delivered at time $t = 3 + 1 = 4$.

5. $order[5] = 4, prep[4] = 3$. This order is delivered at time $t = 4 + 3 = 7$.

When we order these by ascending fulfillment time, we get:

- $t = 4$: customer 4.
- $t = 6$: customer 2.
- $t = 7$: customer 5.
- $t = 9$: customer 1.
- $t = 11$: customer 3.

We print the ordered numbers in the bulleted listed above as `4 2 5 1 3`.

Note: While not demonstrated in these sample cases, recall that any orders fulfilled at the same time must be listed by ascending order number.

Beautiful Pairs

HackerRank

You are given two arrays, A and B, both containing N integers.

A pair of indices (i, j) is *beautiful* if the i^{th} element of array A is equal to the j^{th} element of array B. In other words, pair (i, j) is *beautiful* if and only if $A[i] = B[j]$. A set containing beautiful pairs is called a *beautiful set*.

A beautiful set is called *pairwise disjoint* if for every pair $(l[i], r[i])$ belonging to the set there is no repetition of either $l[i]$ or $r[i]$ values. For instance, if $A = [10, 11, 12, 5, 14]$ and $B = [8, 9, 11, 11, 5]$ the beautiful set $[(1, 2), (1, 3), (3, 4)]$ is not pairwise disjoint as there is a repetition of 1, that is $l[0][0] = l[1][0]$.

Your task is to change **exactly** 1 element in B so that the size of the pairwise disjoint beautiful set is maximum.

Function Description

Complete the *beautifulPairs* function in the editor below. It should return an integer that represents the maximum number of pairwise disjoint beautiful pairs that can be formed.

beautifulPairs has the following parameters:

- *A*: an array of integers
- *B*: an array of integers

Input Format

The first line contains a single integer n, the number of elements in A and B.
The second line contains n space-separated integers $A[i]$.
The third line contains n space-separated integers $B[i]$.

Constraints

- $1 \leq n \leq 10^3$
- $1 \leq A[i], B[i] \leq 10^3$

Output Format

Determine and print the maximum possible number of pairwise disjoint beautiful pairs.

Note: You must first change 1 element in B, and your choice of element must be optimal.

Sample Input 0

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

Sample Output 0

```
4
```

Explanation 0

You are given $A = [1, 2, 3, 4]$ and $B = [1, 2, 3, 3]$.
The beautiful set is $[(0, 0), (1, 1), (2, 2), (2, 3)]$ and maximum sized pairwise disjoint beautiful set is either $[(0, 0), (1, 1), (2, 2)]$ or $[(0, 0), (1, 1), (2, 3)]$.
We can do better. We change the 3^{rd} element of array B from 3 to 4. Now new B array is:
$B = [1, 2, 4, 3]$ and the pairwise disjoint beautiful set is $[(0, 0), (1, 1), (2, 3), (3, 2)]$. So, the answer is 4.
Note that we could have also selected index 3 instead of index 2 but it would have yeilded the same result. Any other choice of index is not optimal.

Sample Input 1

```
6
3 5 7 11 5 8
5 7 11 10 5 8
```

Sample Output 1

```
6
```

Greedy Florist

HackerRank

A group of friends want to buy a bouquet of flowers. The florist wants to maximize his number of *new* customers and the money he makes. To do this, he decides he'll multiply the price of each flower by the number of that customer's previously purchased flowers plus 1. The first flower will be original price, $(0 + 1) \times \textbf{original price}$, the next will be $(1 + 1) \times \textbf{original price}$ and so on.

Given the size of the group of friends, the number of flowers they want to purchase and the original prices of the flowers, determine the minimum cost to purchase all of the flowers. The number of flowers they want equals the length of the c array.

Example
$c = [1, 2, 3, 4]$
$k = 3$

The length of $c = 4$, so they want to buy 4 flowers total. Each will buy one of the flowers priced $[2, 3, 4]$ at the original price. Having each purchased $x = 1$ flower, the first flower in the list, $c[0]$, will now cost $(\textit{current purchase} + \textit{previous purchases}) \times c[0] = (1 + 1) \times 1 = 2$. The total cost is $2 + 3 + 4 + 2 = 11$.

Function Description

Complete the *getMinimumCost* function in the editor below.

getMinimumCost has the following parameter(s):

- *int c[n]:* the original price of each flower
- *int k:* the number of friends

Returns
- *int:* the minimum cost to purchase all flowers

Input Format

The first line contains two space-separated integers n and k, the number of flowers and the number of friends.
The second line contains n space-separated positive integers $c[i]$, the original price of each flower.

Constraints

- $1 \leq n, k \leq 100$
- $1 \leq c[i] \leq 10^6$
- $answer < 2^{31}$
- $0 \leq i < n$

Sample Input 0

```
3 3
2 5 6
```

Sample Output 0

```
13
```

Explanation 0

There are $n = 3$ flowers with costs $c = [2, 5, 6]$ and $k = 3$ people in the group. If each person buys one flower, the total cost of prices paid is $2 + 5 + 6 = 13$ dollars. Thus, we print 13 as our answer.

Sample Input 1

```
3 2
2 5 6
```

Sample Output 1

```
15
```

Explanation 1

There are $n = 3$ flowers with costs $c = [2, 5, 6]$ and $k = 2$ people in the group. We can minimize the total purchase cost like so:

1. The first person purchases 2 flowers in order of decreasing price; this means they buy the more expensive flower ($c_1 = 5$) first at price $p_1 = (0 + 1) \times 5 = 5$ dollars and the less expensive flower ($c_0 = 2$) second at price $p_0 = (1 + 1) \times 2 = 4$ dollars.
2. The second person buys the most expensive flower at price $p_2 = (0 + 1) \times 6 = 6$ dollars.

We then print the sum of these purchases, which is $5 + 4 + 6 = 15$, as our answer.

Sample Input 2

```
5 3
1 3 5 7 9
```

Sample Output 2

```
29
```

Explanation 2

The friends buy flowers for 9, 7 and 3, 5 and 1 for a cost of $9 + 7 + 3 * (1 + 1) + 5 + 1 * (1 + 1) = 29$.