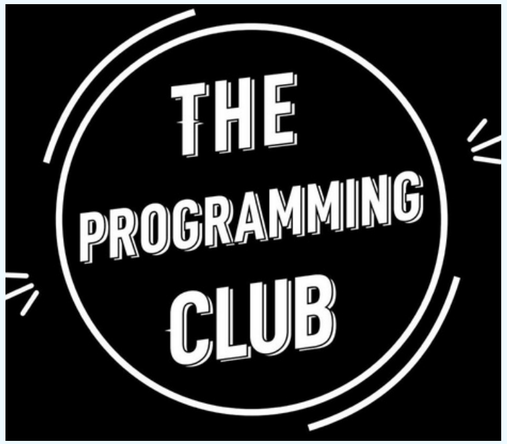


PREFIX SUMS AND STRINGS

Let us start the session with a question

Let there be an n integers given in a certain order .

Say we are given q Queries where the ith query is Qi : = (a,b) , a pair of integers.
For each query it is required to find the sum of all elements from ath element to bth element. q is on the third line of input. And the q queries are in the next q lines of input

How would we solve this??

The first approach would be to find the sum separately for every query.

Example Test Case

Example input	Expected Output
7 (number of integers)	14
1 4 3 6 1 5 8	27
3 (number of queries)	6
1 4	
2 7	
5 6	

Why do we need prefix sum?

If we do it inefficiently like that how much computation will we be doing in worst case ?

Suppose we spend a Coin per operation.

```
int n;
cin >> n;
int arr[n];
for(int i = 0; i < n; i++){
    cin>> arr[i];
}
int q;
cin > q;
for(int i 0; i < q; i++){
    int a , b, sum = 0;
    cin >> a >> b;
    for(int j = a;j <= b;j++){
        sum = sum + arr[i];
    }
    cout << sum <<"\n";
}
```

How many coins ?

If we calculate the sum of elements of each query separately we will be spending some approximately n*q coins .

At worst case , a is near start of array , and b is near end of array so we will spend nearly n coins per query .

There are q queries . Hence n*q coins .

Also input , output for each query which adds some kq coins .
Major term is n*q when n is large , k is small constant .

How to find the prefix sum array and what is it?

Define Prefix Sum array : Prefix sum array of a set of n integers / float / ... consists of n+1 elements .

We would define the prefix sum array as

```
int ps[n+1];
```

Where ps[i] would be the **sum of first i elements** of the array of n integers .

And ps[0] is defined as zero.

What is the **recursive relation** here ? BASED on the way we have defined prefix sum

How to find the prefix sum array?

We can observe that ps[k]=ps[k-1]+arr[k-1]

```
int ps[n+1];
ps[0] = 0;

//ps[i] is the sum of the first i elements of arr

for(int i=1;i<n+1;i++){
    ps[i] = ps[i-1] + arr[i-1];
}
```

Prefix Sum Method

We initialize ps[0]=0

For o<k<n+1 ps[k] = arr[0]+arr[1]+arr[2]+...+arr[k-1]

Now for each query

arr[a-1]+arr[a+1]+...+arr[b-1] = arr[0]+arr[1]+...+arr[b-1]-(arr[0]+arr[1]+...+arr[a-2])

The required sum for each query = ps[b]-ps[a-1]

We are spending just one coin per query nearly .

And we spent about n coins to find prefix sum array .

Total coins spent n + q .

Applications of Prefix Sums

01

Sum of elements of subarray

02

Finding Maximum Sum of k element sub array

03

Finding Maximum Subarray Sum

i) Finding subarray sum

We can find the sum of all elements from l th element to r th element efficiently once we have calculated all prefix sums .

```
Sum_from_l_to_r = ps[r] - ps[l-1];
```

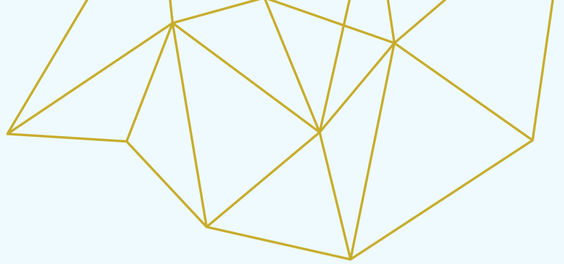
ii) Finding the subarray containing k elements and with maximum sum

This is a more efficient approach than using nested loops.

```
int max = ps[k];

for(int i = 1 ; i < = n-k ; i++){
    int sum = ps[i + k] - ps[i];
    if(sum > max){
        max = sum;
    }
}

cout << max;
```



Try it out later !!!!

iii)Finding maximum subarray sum

The maxs variable is updated by comparing the current prefix sum minus minimum prefix sum so far and current maxs. This helps us to find the maximum sum so far.It effectively considers all possible subarrays and selects the one with maximum sum.
minps stores the minimum prefixsum encountered so far.

```
maxsum=INT_MIN;
minps=0;

for(int i=0;i<n;i++){
    minps=min(minps,ps[i]);
    maxsum=max(maxsum,ps[i]-minps);
}
cout<<maxsum;
```

B. Promo

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The store sells n items, the price of the i-th item is p_i. The store's management is going to hold a promotion: if a customer purchases at least x items, y cheapest of them are free.

The management has not yet decided on the exact values of x and y. Therefore, they ask you to process q queries: for the given values of x and y, determine the maximum total value of items received for free, if a customer makes **one purchase**.

Note that all queries are independent; they don't affect the store's stock.

Input

The first line contains two integers n and q ($1 \le n, q \le 2 \cdot 10^5$) — the number of items in the store and the number of queries, respectively.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^6$), where p_i — the price of the i-th item.

The following q lines contain two integers x_i and y_i each ($1 \le y_i \le x_i \le n$) — the values of the parameters x and y in the i-th query.

Output

For each query, print a single integer — the maximum total value of items received for free **for one purchase**.

Example

input

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

output

```
8
5
6
```

```cpp
int n,q;
cin >> n >> q;

int p[n];
for(int i=0;i<n;i++){
    cin >> p[i];
}

sort(p,p+n);
int ps[n+1];

ps[0]=0;
for(int i=1;i<n+1;i++){
    ps[i] = ps[i-1] + p[i-1];
}

for(int i=0;i<q;i++){
    int x,y;
    cin >> x >> y;
    cout << ps[n-x+y] - ps[n-x]<<"\n";

}
```

This can also be done using inefficient methods but it exceeds the given time limit in that case.

B. Lecture Sleep

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Your friend Mishka and you attend a calculus lecture. Lecture lasts n minutes. Lecturer tells a_i theorems during the i-th minute.

Mishka is really interested in calculus, though it is so hard to stay awake for all the time of lecture. You are given an array t of Mishka's behavior. If Mishka is asleep during the i-th minute of the lecture then t_i will be equal to 0, otherwise it will be equal to 1. When Mishka is awake he writes down all the theorems he is being told — a_i during the i-th minute. Otherwise he writes nothing.

You know some secret technique to keep Mishka awake for k minutes straight. However you can use it **only once**. You can start using it at the beginning of any minute between 1 and $n - k + 1$. If you use it on some minute i then Mishka will be awake during minutes j such that $j \in [i, i + k - 1]$ and will write down all the theorems lecturer tells.

You task is to calculate the maximum number of theorems Mishka will be able to write down if you use your technique **only once** to wake him up.

Input

The first line of the input contains two integer numbers n and k ($1 \le k \le n \le 10^5$) — the duration of the lecture in minutes and the number of minutes you can keep Mishka awake.

The second line of the input contains n integer numbers $a_1, a_2, \dots a_n$ ($1 \le a_i \le 10^4$) — the number of theorems lecturer tells during the i-th minute.

The third line of the input contains n integer numbers $t_1, t_2, \dots t_n$ ($0 \le t_i \le 1$) — type of Mishka's behavior at the i-th minute of the lecture.

Output

Print only one integer — the maximum number of theorems Mishka will be able to write down if you use your technique **only once** to wake him up.

Example

input

```
6 3
1 3 5 2 5 4
1 1 0 1 0 0
```

output

```
16
```

Strings

Inbuilt into c++

Sequence of characters .

Accessed like an array

>> operator considers **whitespace or new line** as terminating while taking string input .

Getline function will take **one full line** as input for string .

```
string str1 ;
cin >> str1;


string str2;
getline(cin , str2);
```

Declaration and Initialisation

```
string str1 = "hello" ;

//OR

string str2;
str2 = "bye"
```

Characters ASCII

Characters are stored as its **ASCII** code in a computer .
Each character is associated with a unique integer ranging from 0 to 255

0	NUL	16	DLE	32		48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

''A' - 'Z' from 65 to 90
''a' - 'z' from 97 to 122
'0' - '9' from 48 to 57

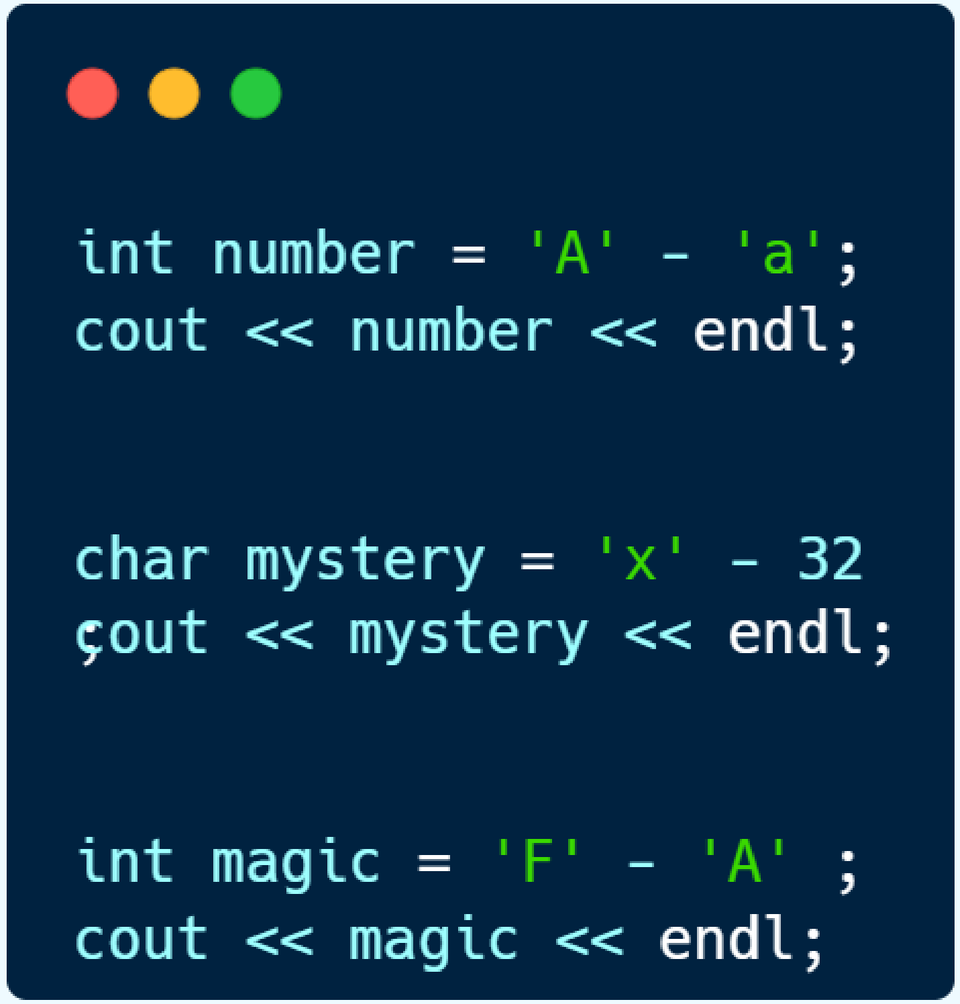
Adding 32 to any uppercase letter converts it to lowercase letter .

Subtracting 32 from lowercase letter concerts it to uppercase letter .

Dec	Hex	Char
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z

Dec	Hex	Char
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z

Dec	Hex	Char
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9

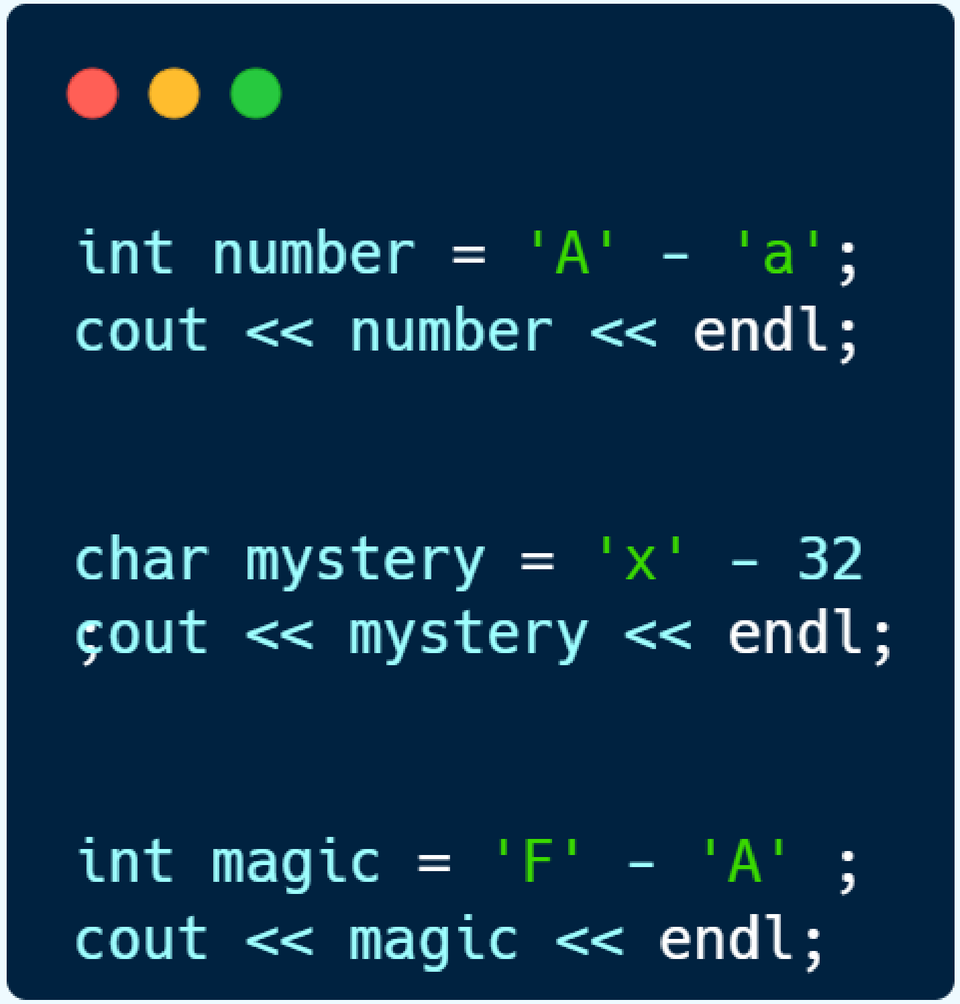


```
int number = 'A' - 'a';
cout << number << endl;


char mystery = 'x' - 32
çout << mystery << endl;


int magic = 'F' - 'A' ;
cout << magic << endl;
```

Output ?



```
int number = 'A' - 'a';
cout << number << endl;


char mystery = 'x' - 32
cout << mystery << endl;


int magic = 'F' - 'A' ;
cout << magic << endl;
```

Output ?

-32

X

5

String functions

clear()

```
string str = "hello" ;
str.clear();
//removes all characters in string ,
cout << str ;
//prints nothing
```

String functions

length()

```
string str = "hello" ;

cout << str.length( ) ;
//prints 5
```

String functions

+ operator

```
string str1 = "bye";
string str2 = " ";
string str3 = "guys";
string message = str1 + str2 + str3;

cout << message ;
//bye guys
```

String functions

[] operator

just like array

```
string str1 = "abcdefgh";

char ch1 = str1[1];
char ch2 = str1[3];

cout << ch1 << endl;
//prints b
cout << ch2 << endl;
//prints d
```

String comparison

Dictionary order based comparison .
Called lexicographical comparison .

```
string str1 = "apple";
string str2 = "apple";
string str3 = "ball";

if(str1 == str2){
  cout << "Same strings" << endl;
}

if(str1 < str3){
  cout << "apple comes before ball in dictionary";
}
```

String functions

Check out append , substr , reverse ,find functions for strings as well online !

Example

Count the number of occurences of letter a in the string and output .

```
string str = "babhaabaaa";

int len = str.length();
int num = 0;

for(int i = 0 ; i < len ; i++)
{
  if(str[i] == 'a') num++;
}

cout << num ;
```