**C++ functions**

**• lower\_bound returns a pointer to the first array element whose value is at**

**least x.**

**• upper\_bound returns a pointer to the first array element whose value is**

**larger than x.**

**• equal\_range returns both above pointers.**

**The functions assume that the array is sorted. If there is no such element,**

**the pointer points to the element after the last array element. For example, the**

**following code finds out whether an array contains an element with value x:**

**auto k = lower\_bound(array,array+n,x)-array;**

**if (k < n && array[k] == x) {**

**// x found at index k**

**}**

**Then, the following code counts the number of elements whose value is x:**

**auto a = lower\_bound(array, array+n, x);**

**auto b = upper\_bound(array, array+n, x);**

**cout << b-a << "\n";**

**Using equal\_range, the code becomes shorter:**

**auto r = equal\_range(array, array+n, x);**

**cout << r.second-r.first << "\n";**

**Binary search first occurrence**

**ll bs(ll l,ll r,ll val,vector<ll>&v)**

**{**

**ll last=-1;**

**while(l<=r)**

**{**

**ll mid=(l+r)/2;**

**if (val<=v[mid])**

**{**

**last=mid;**

**r=mid-1;**

**}**

**else {l=mid+1;}**

**}**

**return last;**

**}**

**Binary search last occurrence**

**ll bs(ll l,ll r,ll val,vector<ll>&v)**

**{**

**ll last=-1;**

**while(l<=r)**

**{**

**ll mid=(l+r)/2;**

**if (val<=v[mid])**

**{**

**last=mid;**

**r=mid-1;**

**}**

**else {l=mid+1;}**

**}**

**return last;**

**}**

**Binary search on function**

C. Sagheer and Nubian Market

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

On his trip to Luxor and Aswan, Sagheer went to a Nubian market to buy some souvenirs for his friends and relatives. The market has some strange rules. It contains *n* different items numbered from 1 to *n*. The *i*-th item has base cost *ai* Egyptian pounds. If Sagheer buys *k*items with indices *x*1, *x*2, ..., *xk*, then the cost of item *xj* is *axj* + *xj*·*k* for 1 ≤ *j* ≤ *k*. In other words, the cost of an item is equal to its base cost in addition to its index multiplied by the factor *k*.

Sagheer wants to buy as many souvenirs as possible without paying more than *S* Egyptian pounds. Note that he cannot buy a souvenir more than once. If there are many ways to maximize the number of souvenirs, he will choose the way that will minimize the total cost. Can you help him with this task?

**Input**

The first line contains two integers *n* and *S* (1 ≤ *n* ≤ 105 and 1 ≤ *S* ≤ 109) — the number of souvenirs in the market and Sagheer's budget.

The second line contains *n* space-separated integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 105) — the base costs of the souvenirs.

**Output**

On a single line, print two integers *k*, *T* — the maximum number of souvenirs Sagheer can buy and the minimum total cost to buy these *k*souvenirs.

**Examples**

**input**

**Copy**

3 11  
2 3 5

**output**

**Copy**

2 11

**input**

**Copy**

4 100  
1 2 5 6

**output**

**Copy**

4 54

**input**

**Copy**

1 7  
7

**output**

**Copy**

0 0

**Note**

In the first example, he cannot take the three items because they will cost him [5, 9, 14] with total cost 28. If he decides to take only two items, then the costs will be [4, 7, 11]. So he can afford the first and second items.

In the second example, he can buy all items as they will cost him [5, 10, 17, 22].

In the third example, there is only one souvenir in the market which will cost him 8 pounds, so he cannot buy it.

him 8 pounds, so he cannot buy it.

**vector<ll>dighere;**

**ll n,s;**

**ll func(ll k)**

**{**

**ll ans=0;**

**vector<ll>digthere(n);**

**for (int i=0;i<n;i++)**

**{**

**digthere[i]=(dighere[i]+((i+1)\*k));**

**}**

**sort(digthere.begin(),digthere.end());**

**for (ll i=0;i<k;i++)**

**{**

**ans+=digthere[i];**

**}**

**return ans;**

**}**

**int main()**

**{**

**//freopen("pizza.in","r",stdin);**

**//freopen("output.txt","w",stdout);**

**ios\_base::sync\_with\_stdio(false);**

**cin.tie(NULL);**

**cin>>n>>s;**

**for (int i=0;i<n;i++)**

**{**

**ll x;**

**cin>>x;**

**dighere.push\_back(x);**

**}**

**ll low=0;**

**ll high=n;**

**while(low<=high)**

**{**

**ll mid=low+high;**

**mid/=2;**

**if (func(mid)<=s){low=mid+1;}**

**else {high=mid-1;}**

**}**

**cout<<low-1<<" "<<func(low-1)<<endl;**

**return 0;**

}

**Binary search on doubles**

**float l = 0.0f, r = 1e14f;**

**for (int i = 0; i < iteration\_count && l + eps < r; ++i)**

**{**

**float mid = 0.5f \* (l + r);**

**if (check(mid))**

**r = mid;**

**else**

**l = mid;**

**}**

**double** a = 0.0; **//never gonna happend**

**double** r = 100000000000000000000.0; **//never gonna happen**

**for** (**int** it = 1; it <= 70; it++) { **// some coders use 40, others 70, Same varies according to the problem, honestly I do not know how it is calculated**

**double** mid = (a + b) / 2.0;

**if** (propertie(mid)) b = mid;

**else** a = mid;

}