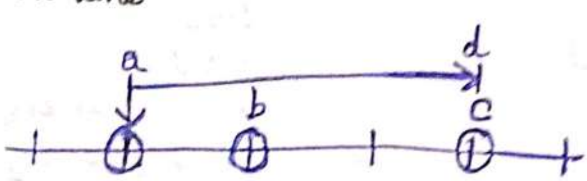
BINARY SEARCH APPLICATION

1. Points on line



For every starting pts we are trying to find possible ways.

For exa :-

4 3

$N=4$
$d=3$

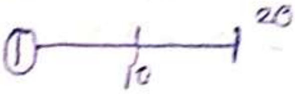
$\underline{1}\ \underline{2}\ 3\ 4$

$\rightarrow {}^3C_2 = 3 + {}^2C_2 = 4$

$(1, -, -) + (2, -, -)$

5 19

1 10 20 30 50



$\Rightarrow {}^1C_2 = 0$

~~Code :-~~

~~void~~ solve ~~(int~~

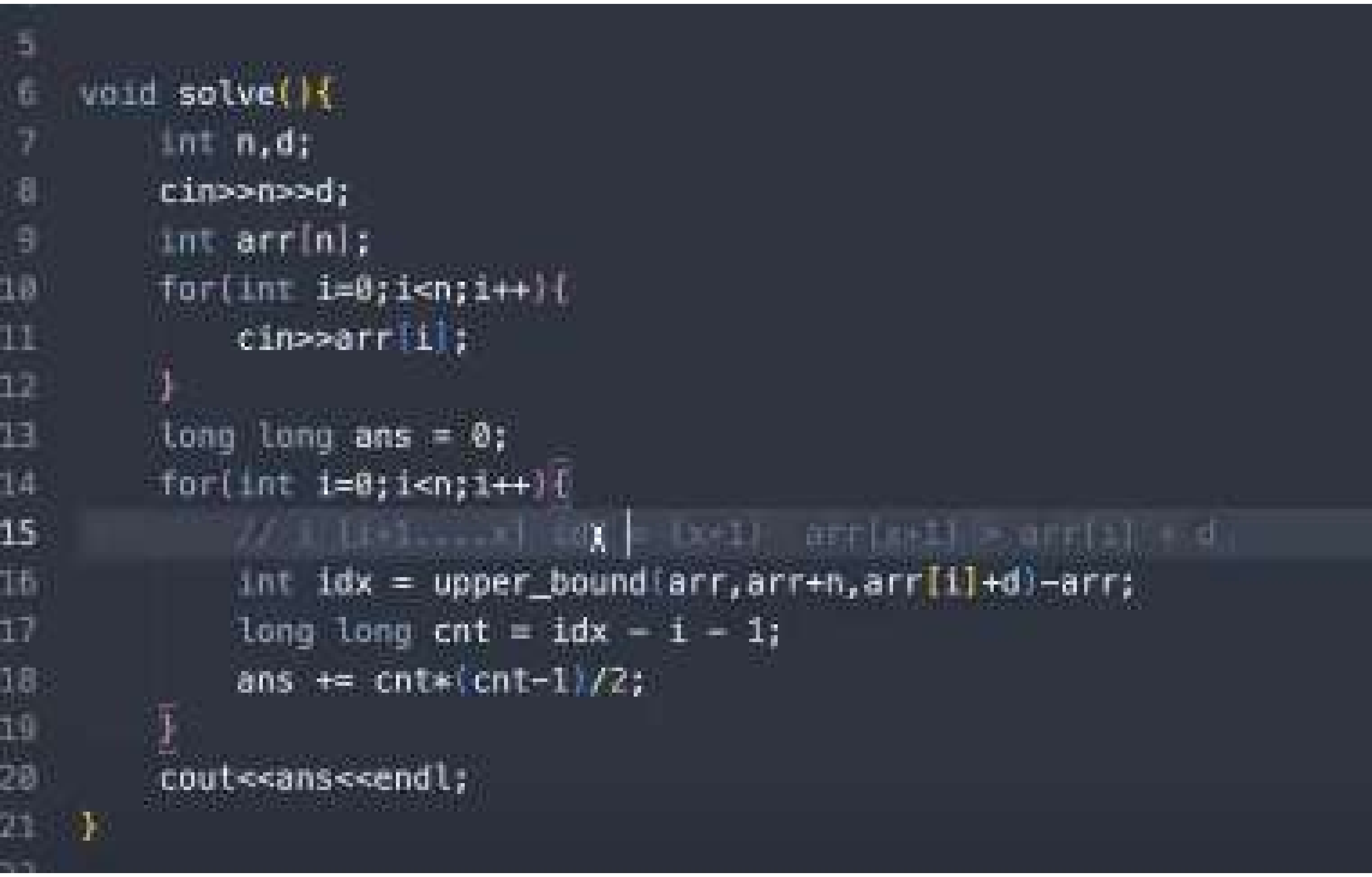
No. of pts. b/w 1 & 4 = 2

So, for starting pt = 1

we have ${}^3C_2$ ways.

For $i \rightarrow$ farthest point $= i+d$

We are finding elements from $i+1 \dots n$ such that $\underbrace{arr[x] <= arr[i]+d}_{\text{check func}^n}$.

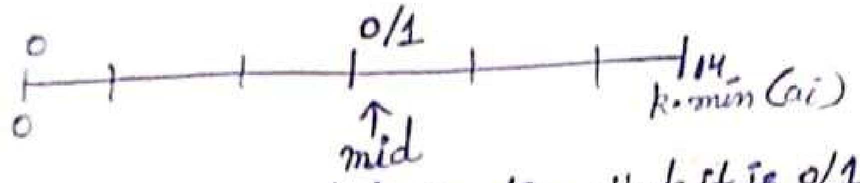


```
void solve(){
    int n,d;
    cin>>n>>d;
    int arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    long long ans = 0;
    for(int i=0;i<n;i++){
        // i [i+1....x] idx = (x+1)  arr[x+1] > arr[i] + d
        int idx = upper_bound(arr,arr+n,arr[i]+d)-arr;
        long long cnt = idx - i - 1;
        ans += cnt*(cnt-1)/2;
    }
    cout<<ans<<endl;
}
```

2. Factory Machines * This is BS on Ans.

Search space → time
limits low = 0, high = k·min(ai)

check [~~mid~~ x] = can we generate K products in X time



If there is a check function that it is 0/1, we can find starting pt. using binary search.

$M_1 \rightarrow a_1 \rightarrow \frac{x}{a_i} \rightarrow$ No. of products that machine i will generate in x time.

$\boxed{lo = 0}$ $\boxed{hi = 10^{18}}$

$\therefore \boxed{\sum \left[\frac{x}{a_i}\right] \geq k}$ → 1
→ 0
→ Total products

First feasible time → optimal time

We have to check that if this total $\left(\sum\left[\frac{x}{a_i}\right]\right) \geq$ K. If yes → 1 else 0.

3. Painter Partition

search space → time

check func^n : Can we paint in x time using ≤ k painter

Logic :

→ If k painters can finish the job within time mid, ans = mid.
& update ~~high = low~~ upper bound = mid-1.

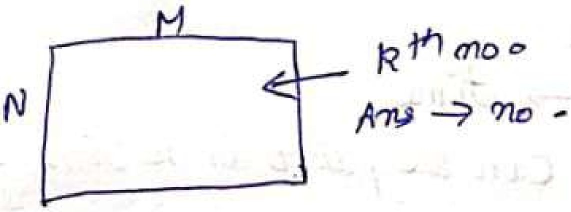
→ If more than k painters needed, lo = mid+1.

Code :-

```
int check (int x) {
    int painter = 0;
    int timeleft = 0;
    for (int i = 0; i<n; i++) {
        if (timeleft >= arr[i]) {
            timeleft -= arr[i]; }
        else {
            painter ++;
            timeleft = x-arr[i];
    }
}
```

```
    if (painter <=k) return 1;
    else return 0;
}
void solve ( ) {
    cin >> n >> k;
    int maxlen = 0;
    int sum = 0;
    for (int i=0 ; i<n; i++) {
        cin >> arr [i];
        maxlen = max (maxlen, arr [i]);
        sum += arr [i];
    }
    int lo = maxlen;
    int hi = sum;
    int ans = -1;
    while (lo <= hi) {
        int mid = (lo+hi)/2;
        if (check (mid)) {
            ans = mid;
            hi = mid-1; }
        else lo = mid+1;
    }
    cout << ans ;
}
```

4. Multiplication Table



For kth value

check (x) : $\boxed{(\text{No. of val} \le x) \ge x}$

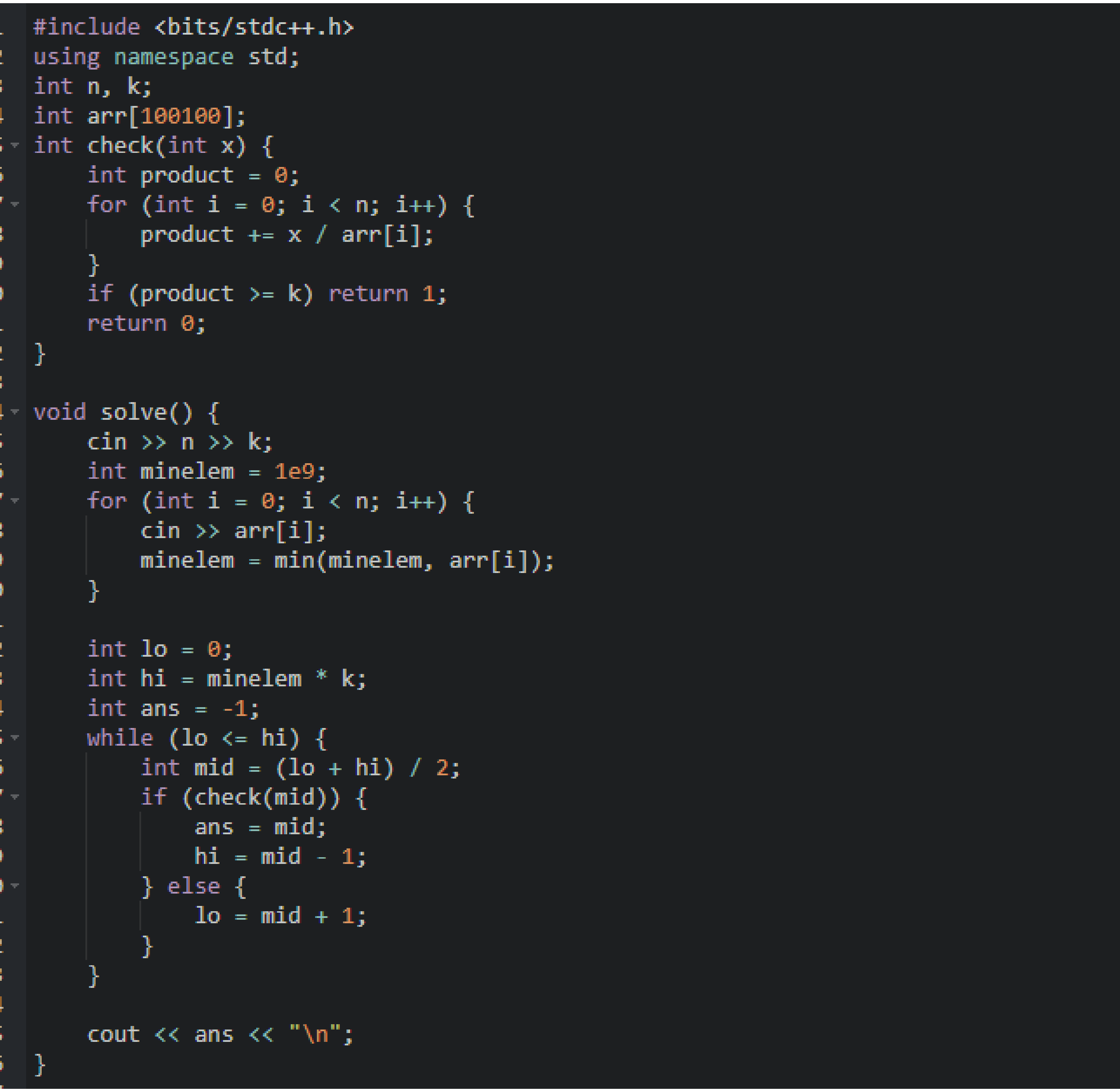
For exa: 2 3 ④ → x=4

1	2	3
2	4	6

1, 2, 2, ③ 4, 6

check(2) = 0
Check (3) = 1
Check (4) = 1

If we find the first ① → that is the ans.

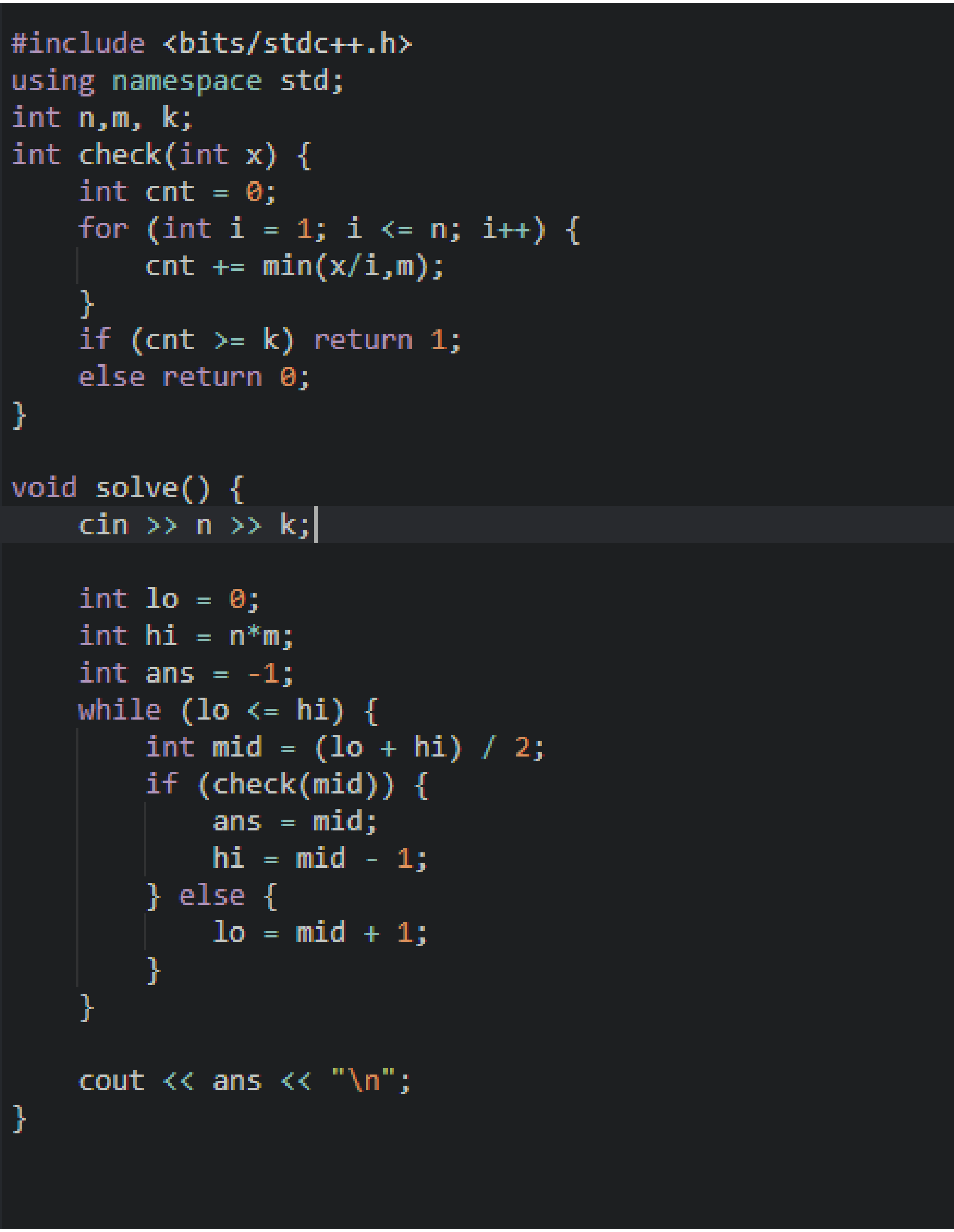


```cpp
#include <bits/stdc++.h>
using namespace std;
int n, k;
int arr[100100];
int check(int x) {
    int product = 0;
    for (int i = 0; i < n; i++) {
        product += x / arr[i];
    }
    if (product >= k) return 1;
    return 0;
}

void solve() {
    cin >> n >> k;
    int minelem = 1e9;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
        minelem = min(minelem, arr[i]);
    }

    int lo = 0;
    int hi = minelem * k;
    int ans = -1;
    while (lo <= hi) {
        int mid = (lo + hi) / 2;
        if (check(mid)) {
            ans = mid;
            hi = mid - 1;
        } else {
            lo = mid + 1;
        }
    }

    cout << ans << "\n";
}
```



```
#include <bits/stdc++.h>
using namespace std;
int n,m, k;
int check(int x) {
    int cnt = 0;
    for (int i = 1; i <= n; i++) {
        cnt += min(x/i,m);
    }
    if (cnt >= k) return 1;
    else return 0;
}

void solve() {
    cin >> n >> k;

    int lo = 0;
    int hi = n*m;
    int ans = -1;
    while (lo <= hi) {
        int mid = (lo + hi) / 2;
        if (check(mid)) {
            ans = mid;
            hi = mid - 1;
        } else {
            lo = mid + 1;
        }
    }

    cout << ans << "\n";
}
```