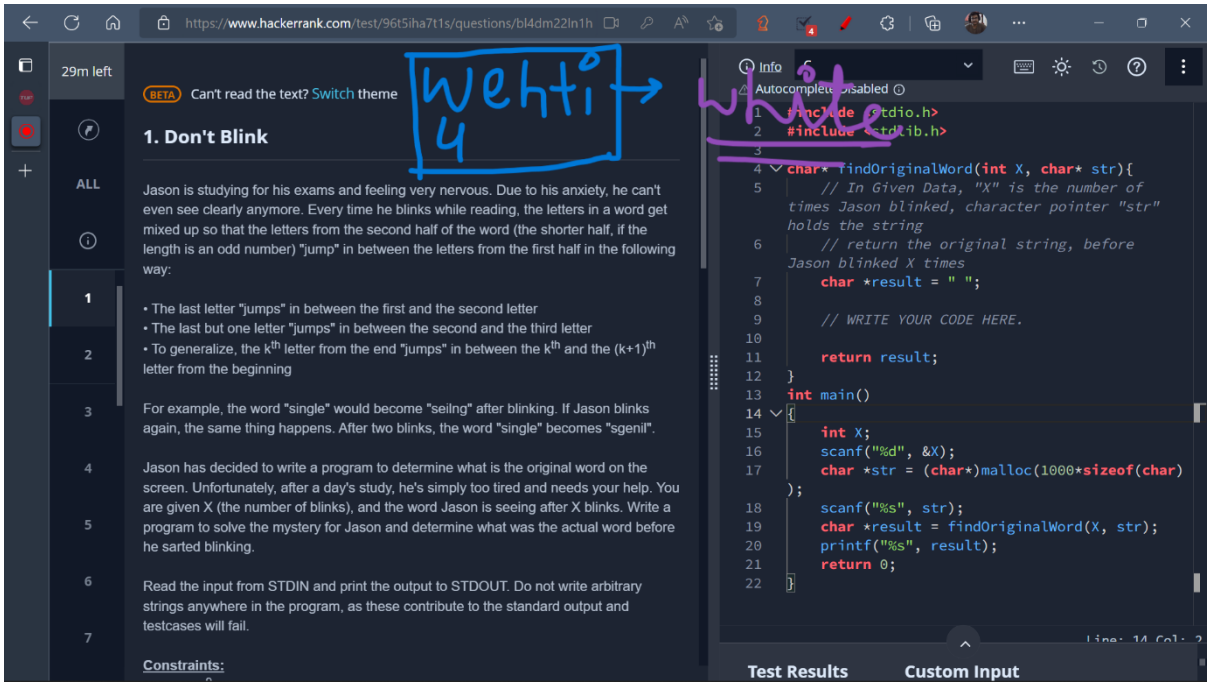
Goldman Sachs

1)



2)

2. Efficient Queuing

An Indian engineers' delegation, consisting of P people, is traveling to the International Engineering Fair, organized in New York. They are currently waiting in a queue for check-in at the airport. There are N check-in desks open. Some check-in officials work more efficiently than others, so the desks operate at different speeds. At the k-th desk, T_k minutes are required to finish the check-in of a single passenger, and members of our delegation happen to know the exact times for each desk.

In the beginning, all desks are ready to accept the next passenger, and the delegation members are the only people in the queue. A person can only start check-in at an available desk when all people in front of that person in the queue have already started check-in. When it is their turn, a person can immediately occupy an available desk, or can choose to wait for another desk to become available. Our delegation members, being science geeks, make this decision in such a way that the moment when all of them finish check-in is as fast as possible. Your task is finding that moment of time.

Read the input from STDIN and print the output to STDOUT. Do not write arbitrary strings while reading the input or while printing, as these contribute to the standard output.

Constraints:

$1 \le N \le 100,000$

$1 \le P \le 10^9$

$1 \le T_k \le 10^9$

Input Format:

The first line of input contains two space-separated integers, N, the number of desks, and P, the number of people in the delegation.

The second line of input contains N space-separated integers, denoting time taken at each desk T_k, where $1 <= k <= N$.

Output Format:

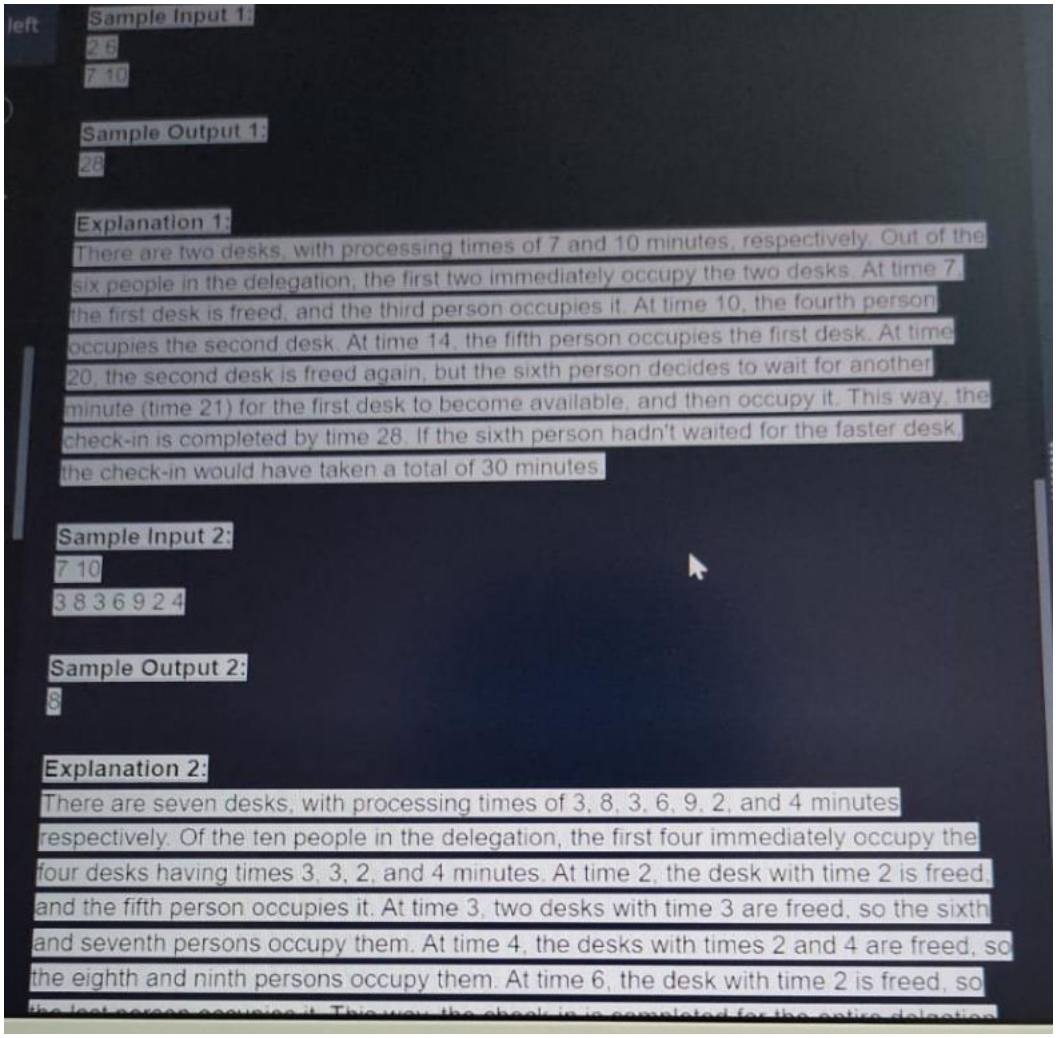
Single line of output contains the fastest time when the entire delegation can complete check-in.

Sample Input 1:

```
2 6
7 10
```

Sample Output 1:

```
28
```


left
Sample Input 1:
2 6
7 10
Sample Output 1:
28
Explanation 1:
There are two desks, with processing times of 7 and 10 minutes, respectively. Out of the six people in the delegation, the first two immediately occupy the two desks. At time 7, the first desk is freed, and the third person occupies it. At time 10, the fourth person occupies the second desk. At time 14, the fifth person occupies the first desk. At time 20, the second desk is freed again, but the sixth person decides to wait for another minute (time 21) for the first desk to become available, and then occupy it. This way, the check-in is completed by time 28. If the sixth person hadn't waited for the faster desk, the check-in would have taken a total of 30 minutes.
Sample Input 2:
7 10
3 8 3 6 9 2 4
Sample Output 2:
8
Explanation 2:
There are seven desks, with processing times of 3, 8, 3, 6, 9, 2, and 4 minutes respectively. Of the ten people in the delegation, the first four immediately occupy the four desks having times 3, 3, 2, and 4 minutes. At time 2, the desk with time 2 is freed, and the fifth person occupies it. At time 3, two desks with time 3 are freed, so the sixth and seventh persons occupy them. At time 4, the desks with times 2 and 4 are freed, so the eighth and ninth persons occupy them. At time 6, the desk with time 2 is freed, so

Solutions:

Atlassian Graph Question→

```
void swap(int &a, int &b){
    int c = a;
    a = b;
    b = c;
}

struct DSU{
    int n;
    vector<int> parent;
    vector<int> size, rank;

    DSU(int n){
        this -> n = n;
        for(int i = 0; i <= n; i++){
            parent.push_back(i);
            size.push_back(1);
            rank.push_back(i);
        }
    }

    int find_parent(int v){
        if(parent[v] == v)
            return v;
        return parent[v] = find_parent(parent[v]);
    }

    int unite(int a, int b){
        a = find_parent(a);
        b = find_parent(b);
```

```
        if(a == b)
            return 0;
        int least = min(rank[a], rank[b]);
        if(size[a] < size[b])
            swap(a, b);
        parent[b] = a;
        size[a] += size[b];
        rank[a] = max(rank[a], rank[b]);
        return least;
    }
};

vector<long> solve(int c_nodes, vector<int> c_from, vector<int> c_to){

    long curr = (c_nodes * (c_nodes + 1)) / 2;
    DSU dsu(c_nodes);
    vector<long> ans;
    int m = c_from.size();
    for(int i = 0; i < m; i++){
        curr -= dsu.unite(c_from[i], c_to[i]);
        ans.push_back(curr);
    }
    return ans;
}
```

Goldman Sachs Solutions →

1)

```
string converter(string &s){
   int n = ln(s);
   string st = "", tt = "";

   if(n&1) {
      st = s.substr(n/2+1, n/2);
      tt = s.substr(0, n/2+1);
   }
   else {
      st = s.substr(n/2, n/2);
      tt = s.substr(0, n/2);
   }

   string final = "";
   int i = ln(st)-1, j = 0;
   while(i >= 0 && j < ln(tt)){
      final += tt[j];
      final += st[i];
      i--;
      j++;
   }
   if(j < ln(tt)) final += tt[j];
   return final;
}

string findOriginalWord(int x, string s) {
   int n = ln(s);
   if(n == 1 || n == 2) return s;
```

```
    string t = converter(s);
    int moves = 1;
    while(1){
        if(s == t) break;
        moves++;
        t = converter(t);
    }
    x = x%moves;
    x = moves-x;
    while(x--) s = converter(s);
    return s;
}
```

2)

```
ll minimumTime(int N, int M, int T[]) {
    ll n, p;
    n = N;
    p = M;
    ll arr[n];

    for(int i=0 ;i <n; i++) arr[i] = T[i];

    ll lo = 0, hi = 1e18;

    auto check = [&](ll num) -> bool{
        ll done = 0;
        for(int i=0; i<n; i++) done += num / arr[i];
```

```
        return done >= p;
    };

    while(lo <= hi){
        ll mid = lo + (hi - lo) / 2;
        if(check(mid))
            hi = mid - 1;
        else
            lo = mid + 1;
    }
    return (hi+1);
}
```