Solution 1:-

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
    int E, N, count = 0;
    cin>>E>>N;
    vector<int> exer(N);
    for(int i = 0; i < N; i++)
        cin>>exer[i];

    sort(exer.rbegin(), exer.rend());
    for(int i = 0; i < N; i++)
    {
        if(E <= 0) break;
        E = E - exer[i];
        count++;
        if(E <= 0) break;
        E = E - exer[i];
        count++;
    }
    if(E <= 0)
        cout<<count<<endl;
    else
        cout<<-1<<endl;
}
```

Solution 2:-

```
#include <iostream>
#include<vector>

using namespace std;

int main()
{
```

```
    int n, m, H, i;
    cin>>n>>m>>H;
    int vptr = n-1, hptr = 0;
    vector<int> villains(n);

    for(i=0; i<n; i++)
        cin>>villains[i];

    while(vptr>=0 && hptr<m) {
        int hpower = H;
        while(vptr>=0) {
            int vpower = villains[vptr];

            if(hpower > vpower) {
                hpower -= vpower;
                vptr--;
            }
            else if(hpower < vpower) {
                hptr++;
                break;
            }
            else {
                vptr--;
                hptr++;
                break;
            }
        }
    }

    if(vptr>=0) cout<<vptr+1;
    else cout<<0;

    return 0;
}
```

Solution 3:-

```
#include <iostream>
#include <vector>
```

```cpp
#include <algorithm>
#include <cmath>

using namespace std;

int main()
{
    int n;
    cin>>n;
    vector<int> arr(n);
    for(int i = 0; i < n; i++)
        cin>>arr[i];

    int prev = arr[0], sum = 0, max = 0, i;

    for(i = 1; i < n; i++)
    {
        if (arr[i] <= prev)
        {
            prev = arr[i];
        }
        else{
            sum = arr[i] - (prev-1);
            int k = floor(sqrt(sum + 2));
            if(k * k == sum + 2)
            {
                sum++;
                prev--;
            }
            prev--;
            max = max>sum?max:sum;
        }
    }
    cout<<ceil(sqrt(max))<<endl;
}
```

Solution 4:-

```cpp
#include <iostream>
```

```cpp
#include <unordered_map>
#include <vector>
using namespace std;

int main(){
    int n;
    cin>>n;
    vector<int> arr(n);

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

    unordered_map<int, int> m;
    if(n&1){
        int mid = n/2;
        for(int i = 0; i < mid; i++){
            if(m.find(arr[i] - i) == m.end()){
                m[arr[i] - i] = 0;
            }
            m[arr[i] - i]++;
        }
        for(int i = mid; i < n; i++){
            if(m.find(arr[i] + i - n + 1) == m.end()){
                m[arr[i] + i - n + 1] = 0;
            }
            m[arr[i] + i - n + 1]++;
        }

    }
    else{
        int mid1 = n/2-1, mid2 = n/2;
        for(int i = 0; i <= mid1; i++){
            if(m.find(arr[i] - i) == m.end()){
                m[arr[i] - i] = 0;
            }
            m[arr[i] - i]++;
        }
        for(int i = mid2; i < n; i++){
            if(m.find(arr[i] + i - n + 1) == m.end()){
```

```
                m[arr[i] + i - n + 1] = 0;
            }
            m[arr[i] + i - n + 1]++;
        }
    }
    int maxval = 0, maxfreq = 0;
    for(auto it: m){
        if(it.second > maxfreq) {
            maxval = it.first;
            maxfreq = it.second;
        }
    }
    int ans = n - maxfreq;
    cout<<ans<<endl;
}
```

Solution 5:-

```
#include <iostream>
#include <vector>
#include <unordered_map>

using namespace std;
int gcd(int a, int b)
{
    if(b == 0) return a;
    gcd(b,a%b);
}
int main()
{
    string s;
    cin>>s;
    vector<int> freq(26);

    int n = s.length();
    int ans;
    for(int i = 0; i < n; i++)
    {
        freq[s[i] - 'a'] ++;
```

```
        ans = freq[s[i] - 'a'];
    }

    for(int i = 0; i < 26; i++)
    {
        if(freq[i])
            ans = gcd(ans, freq[i]);
    }
    cout<<ans<<endl;

}
```

Solution 6:-

```
#include <iostream>
#include <vector>
#include <climits>
using namespace std;

struct SegmentTreeNode {
    int value;
    int index;
};

class SegmentTree {
public:
    SegmentTree(const vector<int>& arr) {
        n = arr.size();
        tree.resize(4 * n);
        build(arr, 0, 0, n - 1);
    }

    SegmentTreeNode query(int l, int r) {
        return query(0, 0, n - 1, l, r);
    }

private:
    int n;
```

```
    vector<SegmentTreeNode> tree;

    void build(const vector<int>& arr, int node, int start,
int end) {
        if (start == end) {
            tree[node] = {arr[start], start};
        } else {
            int mid = (start + end) / 2;
            build(arr, 2 * node + 1, start, mid);
            build(arr, 2 * node + 2, mid + 1, end);
            tree[node] = minNode(tree[2 * node + 1], tree[2
* node + 2]);
        }
    }

    SegmentTreeNode query(int node, int start, int end, int
l, int r) {
        if (r < start || end < l) return {INT_MAX, -1};
        if (l <= start && end <= r) return tree[node];
        int mid = (start + end) / 2;
        SegmentTreeNode left = query(2 * node + 1, start,
mid, l, r);
        SegmentTreeNode right = query(2 * node + 2, mid + 1,
end, l, r);
        return minNode(left, right);
    }

    // Modified to prefer farthest (larger index) on tie
    SegmentTreeNode minNode(SegmentTreeNode a,
SegmentTreeNode b) {
        if (a.value < b.value) return a;
        if (a.value > b.value) return b;
        return a.index > b.index ? a : b;  // Prefer farther
index
    }
};

int main() {
```

```
    int N, K;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; ++i) cin >> A[i];
    cin >> K;

    SegmentTree st(A);

    for (int i = 0; i < N; ++i) {
        int right = min(N - 1, i + K);
        if (i + 1 > right) continue;
        SegmentTreeNode minInRange = st.query(i + 1, right);
        if (minInRange.value < A[i]) {
            swap(A[i], A[minInRange.index]);
            break; // Only one swap allowed
        }
    }

    for (int x : A) cout << x << " ";
    cout << endl;
    return 0;
}
```

Solution 7:-

```
#include <iostream>
#include <unordered_map>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int n;
    cin>>n;
```

```
vector<int> arr(n);
unordered_map<int,int> m;
for(int i = 0; i < n; i++)
    cin>>arr[i];

for(int i = 0; i < n; i++)
{
    if(m.find(arr[i]) == m.end())
    {
        m[arr[i]] = 0;
    }
    m[arr[i]] ++;
}
vector<int> freq;
for(auto it:m) freq.push_back(it.second);

sort(freq.begin(), freq.end());

int m1 = freq.size(), maxCount = 0, count = 0;
int k = m1-1;
if(freq[k] & 1){
    maxCount = freq[k];
    freq[k]--;
}
for(int i = freq[k]; i > 0; i--){
    int j = i;
    count = j;
    j /=2;
    k = m1-2;
    while(k>=0)
    {
        if(freq[k] >= j)
        {
            count += j;
        }
        if(j&1) break;
        else j /= 2;
        k--;
```

```
        }
        maxCount = maxCount < count? count:maxCount;
    }
    cout<<maxCount<<endl;
}
```