DCC Club Recruitment Preparation.

Time limit per test: 2 Seconds.

Memory limit per test: 256 Megabytes.

Problem Statement:

The Developers & Coders Club (DCC) at NIT Agartala is hosting a recruitment drive for second-year students. In Round 2 of the recruitment process, students are given a limited amount of preparation time ‘p’ to complete the given tasks. The preparation involves `'n'` tasks, each requiring a specific amount of time. Students want to start preparing from any task ‘i’ and continue with consecutive tasks until they either use up all their preparation time or complete all tasks.

Given the number of tasks `n` and the total preparation time `'p'` available, along with the time required for each task `t1, t2, ..., tn`, determine the maximum number of tasks a student can complete within their available preparation time.

Input:

- The first line contains two integers `n` and `p` (1 ≤ n ≤ 10^5; 1 ≤ p ≤ 10^9) — the number of tasks and the total preparation time available.
- The second line contains a sequence of `n` integers `t1, t2, ..., tn` (1 ≤ ti ≤ 100), where `ti` is the time required for the `i`-th task.

Output:

- Print a single integer — the maximum number of tasks a student can complete.

Examples:

Input:

```
6 10

2 3 1 4 2 5
```

Output:

```
4
```

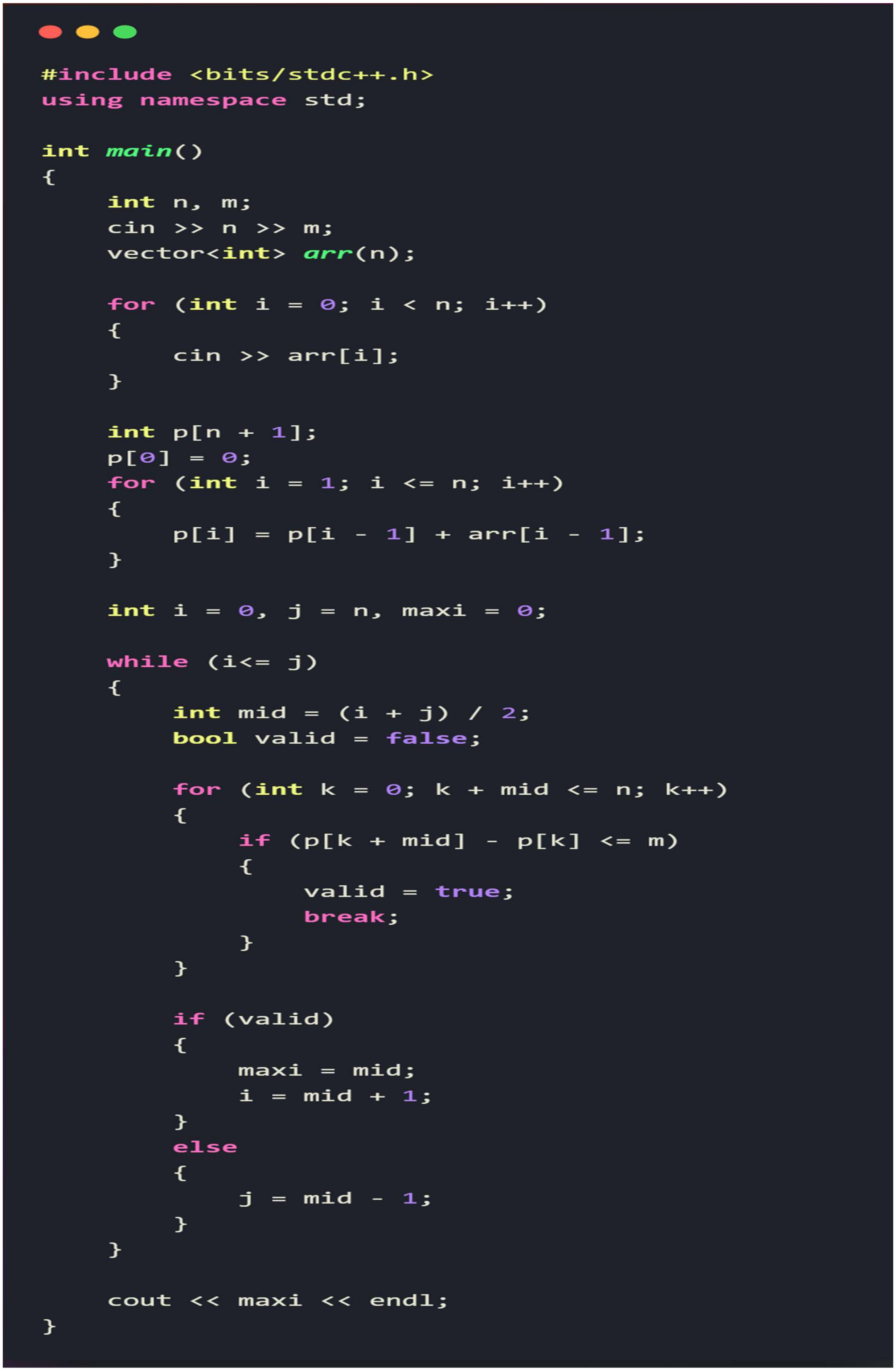
Input:

```
5 7

1 2 2 3 1
```

Output:

```
3
```



```cpp
#include <bits/stdc++.h>
using namespace std;

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

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

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

    int i = 0, j = n, maxi = 0;

    while (i<= j)
    {
        int mid = (i + j) / 2;
        bool valid = false;

        for (int k = 0; k + mid <= n; k++)
        {
            if (p[k + mid] - p[k] <= m)
            {
                valid = true;
                break;
            }
        }

        if (valid)
        {
            maxi = mid;
            i = mid + 1;
        }
        else
        {
            j = mid - 1;
        }
    }

    cout << maxi << endl;
}
```