```
int main()
{
    int e;//Number of eggs
    int f;//Number of floors

    cout<<"Egg dropping puzzle\n\nNumber of eggs:";

    cin>>e;

    cout<<"\nNumber of floors:";

    cin>>f;

    cout<<solvepuzzle(e,f);

    return 0;
}
```

Working with Binomials

Before advancing to the next section, we must see some useful mathematical relations related to binomials.

We know that

$$C_k^n = C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

We also know that the Pascal triangle is

$$\begin{array}{ccccccccccc} & & & & & 1 & & & & & \\ & & & & 1 & & 1 & & & & \\ & & & 1 & & 2 & & 1 & & & \\ & & 1 & & 3 & & 3 & & 1 & & \\ & 1 & & 4 & & 6 & & 4 & & 1 & \\ 1 & & 5 & & 10 & & 10 & & 5 & & 1 \end{array}$$

Pascal (image from Wikipedia)

And we can easily find a recursion if we write the Pascal triangle in this way:

n	k					
	0	1	2	3	4	5
0	1	-	-	-	-	-
1	1	1	-	-	-	-
2	1	2	1	-	-	-
3	1	3	3	1	-	-
4	1	4	6	4	1	-
5	1	5	10	10	5	1

Pascal2

By looking at the table or by a simple mathematical proof we get the following recurrence:

$$C(n,k) = C(n-1,k) + C(n-1,k-1).$$

And the base cases are

$$C(n,0) = \frac{n!}{0!(n-0)!} = 1 \quad \text{and} \quad C(n,n) = \frac{n!}{n!(n-n)!} = 1.$$

A Better Approach

With this knowledge in hand, let's define a function $f(d, n)$ that represents the number of floors we can cover using with d remaining drops. If the egg breaks, we will be able to cover $f(d-1, n-1)$ floors; otherwise we'll be able to $f(d-1, n)$ floors. Hence, the total number of floors we will be able to cover is

$$f(d, n) = 1 + f(d-1, n-1) + f(d-1, n).$$

We must find a function $f(d, n)$ that's a solution for this recursion. First, we will define an auxiliary function $g(d, n)$:

$$g(d, n) = f(d, n+1) - f(d, n).$$

Plugging it into our first equation gives

$$\begin{aligned} g(d, n) &= f(d, n+1) - f(d, n) \\ &= f(d-1, n+1) + f(d-1, n) + 1 - f(d-1, n) - f(d-1, n-1) - 1 \\ &= [f(d-1, n+1) - f(d-1, n)] + [f(d-1, n) - f(d-1, n-1)] \\ &= g(d-1, n) + g(d-1, n-1). \end{aligned}$$

This is precisely the same recursion that we saw in the previous section, and thus the function $g(d, n)$ can be written

$$g(d, n) = \binom{d}{n}.$$

But we have a problem: $f(0, n)$ is 0 for every n, as well as $g(0, n)$, according to the relation between f and g. Howev contradiction occurs when $n = 0$ because $g(0, 0) = \binom{0}{0} = 1$. But $g(0, n)$ should be 0 for every n! We can fix this pr defining $g(d, n)$ as follows:

$$g(d, n) = \binom{d}{n+1}.$$

And the recursion is still valid (you can check it by yourself!).

Now, using a telescopic sum for $f(d, n)$, we can write it as

$$\begin{aligned} f(d, n) =& [f(d, n) - f(d, n-1)] \\ &+ [f(d, n-1) - f(d, n-2)] \\ &+ \cdots \\ &+ [f(d, 1) - f(d, 0)] \\ &+ f(d, 0). \end{aligned}$$

We know that $f(d, 0) = 0$, and therefore

$$f(d, n) = g(d, n-1) + g(d, n-2) + \cdots + g(d, 0).$$

And we also know that

$$g(d, n) = \binom{d}{n+1}.$$

Hence,

$$g(d, n-1) + g(d, n-2) + \cdots + g(d, 0) = \binom{d}{n} + \binom{d}{n-1} + \cdots + \binom{d}{1}.$$

Finally,

$$f(d,n) = \sum_{i=1}^{n} \binom{d}{i}.$$

Now that we have a nice formula for $f(d,n)$, how can we find the minimum number of drops?

It's simple! We know that $f(d,N)$ is the number of floors we can cover in the building with k floors using N eggs an than d drops in the worst cases. We simply have to find a value for d such that

$$f(d,N) \geqslant k.$$

Using our last formula,

$$\sum_{i=1}^{N} \binom{d}{i} \geqslant k.$$

This solution is very fast. We can do a linear search to find a value for d, or we can binary search it for an even faster s

C++ code:

C++

```cpp
#include <iostream>
#include <math.h>

using namespace std;

//Evaluates C(n,k) and verifies if it's greater than or equal to k
long long binomial(int x,int n,int k){

    int i;
    long long int answer=0;
    double aux=1;

    //Calculates C(n,k) using the formula: C(n,k): sum_i_0^k {(n-i+1)/i}
    for(i=1;i<=n;i++){

            aux*=(float)x+1-i;
            aux/=(float)i;
            answer+=aux;

            if(answer>k) break;
    }

    return answer;
}

int main()
{
    int n; //Number of eggs
    int k; //Number of floors

    cout<<"Egg dropping puzzle: ( O(n log k)  solution )\n\n";

    cout<<"Number of floors:";
    cin>>k;

    cout<<"\nNumber of eggs:";
    cin>>n;

    //Binary search variables:
    //Mid: middle
    //Upper: upper limit
    //Inf: inferior limit
```