This is a famous interview question asked in Google, Paytm and many other company interviews.

Below is the problem statement.

```
Imagine you have a special keyboard with the following keys:
Key 1:  Prints 'A' on screen
Key 2: (Ctrl-A): Select screen
Key 3: (Ctrl-C): Copy selection to buffer
Key 4: (Ctrl-V): Print buffer on screen appending it
                 after what has already been printed.

If you can only press the keyboard for N times (with the above four
keys), write a program to produce maximum numbers of A's. That is to
say, the input parameter is N (No. of keys that you can press), the
output is M (No. of As that you can produce).
```

Examples:

```
Input:  N = 3
Output: 3
We can at most get 3 A's on screen by pressing
following key sequence.
A, A, A

Input:  N = 7
Output: 9
We can at most get 9 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V

Input:  N = 11
Output: 27
We can at most get 27 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V, Ctrl A,
Ctrl C, Ctrl V, Ctrl V
```

We strongly recommend to minimize your browser and try this yourself first.

Below are few important points to note.
a) For N < 7, the output is N itself.
b) Ctrl V can be used multiple times to print current buffer (See last two examples above).

The idea is to compute the optimal string length for N keystrokes by using a simple insight. The sequence of N keystrokes which produces an optimal string length will end with a suffix of Ctrl-A, a Ctrl-C, followed by only Ctrl-V's (For N > 6).
The task is to find out the break=point after which we get the above suffix of keystrokes. Definition of a breakpoint is that instance after which we need to only press Ctrl-A, Ctrl-C once and the only Ctrl-V's afterwards to generate the optimal length. If we loop from N-3 to 1 and choose each of these values for the break-point, and compute that optimal string they would produce. Once the loop ends, we will have the maximum of the optimal lengths for various breakpoints, thereby giving us the optimal length for N keystrokes.

Below is C implementation based on above idea.

```c
/* A recursive C program to print maximum number of A's using
   following four keys */
#include<stdio.h>

// A recursive function that returns the optimal length string
// for N keystrokes
int findoptimal(int N)
{
    // The optimal string length is N when N is smaller than 7
    if (N <= 6)
        return N;

    // Initialize result
    int max = 0;

    // TRY ALL POSSIBLE BREAK-POINTS
    // For any keystroke N, we need to loop from N-3 keystrokes
    // back to 1 keystroke to find a breakpoint 'b' after which we
    // will have Ctrl-A, Ctrl-C and then only Ctrl-V all the way.
    int b;
    for (b=N-3; b>=1; b--)
    {
            // If the breakpoint is s at b'th keystroke then
            // the optimal string would have length
            // (n-b-1)*screen[b-1];
            int curr = (N-b-1)*findoptimal(b);
            if (curr > max)
                max = curr;
     }
     return max;
}

// Driver program
int main()
{
    int N;

    // for the rest of the array we will rely on the previous
    // entries to compute new ones
    for (N=1; N<=20; N++)
        printf("Maximum Number of A's with %d keystrokes is %d\n",
               N, findoptimal(N));
}
```

Output:

```
Maximum Number of A's with 1 keystrokes is 1
Maximum Number of A's with 2 keystrokes is 2
Maximum Number of A's with 3 keystrokes is 3
Maximum Number of A's with 4 keystrokes is 4
Maximum Number of A's with 5 keystrokes is 5
Maximum Number of A's with 6 keystrokes is 6
Maximum Number of A's with 7 keystrokes is 9
Maximum Number of A's with 8 keystrokes is 12
Maximum Number of A's with 9 keystrokes is 16
Maximum Number of A's with 10 keystrokes is 20
Maximum Number of A's with 11 keystrokes is 27
Maximum Number of A's with 12 keystrokes is 36
Maximum Number of A's with 13 keystrokes is 48
Maximum Number of A's with 14 keystrokes is 64
Maximum Number of A's with 15 keystrokes is 81
Maximum Number of A's with 16 keystrokes is 108
Maximum Number of A's with 17 keystrokes is 144
Maximum Number of A's with 18 keystrokes is 192
Maximum Number of A's with 19 keystrokes is 256
Maximum Number of A's with 20 keystrokes is 324
```

The above function computes the same subproblems again and again. Recomputations of same subproblems can be avoided by storing the solutions to subproblems and solving problems in bottom up manner.

Below is Dynamic Programming based C implementation where an auxiliary array screen[N] is used to store result of subproblems.

```
/* A Dynamic Programming based C program to find maximum number of A's
   that can be printed using four keys */
#include<stdio.h>

// this function returns the optimal length string for N keystrokes
int findoptimal(int N)
{
    // The optimal string length is N when N is smaller than 7
    if (N <= 6)
        return N;

    // An array to store result of subproblems
    int screen[N];

    int b;  // To pick a breakpoint

    // Initializing the optimal lengths array for uptil 6 input
    // strokes.
    int n;
    for (n=1; n<=6; n++)
        screen[n-1] = n;

    // Solve all subproblems in bottom manner
    for (n=7; n<=N; n++)
    {
        // Initialize length of optimal string for n keystrokes
        screen[n-1] = 0;

        // For any keystroke n, we need to loop from n-3 keystrokes
        // back to 1 keystroke to find a breakpoint 'b' after which we
        // will have ctrl-a, ctrl-c and then only ctrl-v all the way.
        for (b=n-3; b>=1; b--)
        {
            // if the breakpoint is at b'th keystroke then
            // the optimal string would have length
            // (n-b-1)*screen[b-1];
            int curr = (n-b-1)*screen[b-1];
            if (curr > screen[n-1])
                screen[n-1] = curr;
        }
    }

    return screen[N-1];
}

// Driver program
int main()
{
    int N;

    // for the rest of the array we will rely on the previous
    // entries to compute new ones
    for (N=1; N<=20; N++)
        printf("Maximum Number of A's with %d keystrokes is %d\n",
               N, findoptimal(N));
}
```

Output:

```
Maximum Number of A's with 1 keystrokes is 1
Maximum Number of A's with 2 keystrokes is 2
Maximum Number of A's with 3 keystrokes is 3
Maximum Number of A's with 4 keystrokes is 4
Maximum Number of A's with 5 keystrokes is 5
Maximum Number of A's with 6 keystrokes is 6
Maximum Number of A's with 7 keystrokes is 9
Maximum Number of A's with 8 keystrokes is 12
Maximum Number of A's with 9 keystrokes is 16
Maximum Number of A's with 10 keystrokes is 20
Maximum Number of A's with 11 keystrokes is 27
Maximum Number of A's with 12 keystrokes is 36
Maximum Number of A's with 13 keystrokes is 48
Maximum Number of A's with 14 keystrokes is 64
Maximum Number of A's with 15 keystrokes is 81
Maximum Number of A's with 16 keystrokes is 108
Maximum Number of A's with 17 keystrokes is 144
Maximum Number of A's with 18 keystrokes is 192
Maximum Number of A's with 19 keystrokes is 256
Maximum Number of A's with 20 keystrokes is 324
```

Thanks to **Gaurav Saxena** for providing the above approach to solve this problem.