DOUBT SESSION

8. (a--z) ______ / ______ \ ______

- First part: a....z, ~~A....Z~~, 0....9, :
- After "/": a....z, 0....9
- After "\": a--z

Using 2 pointers:

Ex: a / a \ aab

Move this ptr until a--z, 0---9, :

a a : bcc 0 : /

If forward slash then move ahead and check if char is a---z or 1--9.

While (S[i] is lowercase or digit or :)

Ex: aa / 012 \ bb0

\/ ______ / ______ \ ______

No. of alphabets

No. of continuous alphabets

For each corresponding forward slash (/) find backward slash ~~and check the~~,
check the char are a·-- z or 0--- 9 → true
then Before '/' if any '/' or '\' then how many no. of char b/w them.

Approach :-

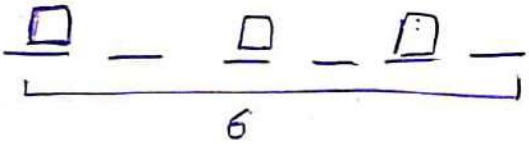
1. Make 5 sets of forward, backward, alphabets, digits and colon to store indices of them when they appear.
2. Apply lowerbound ~~and~~ or upperbound on first forward slash and it will lead us to next backslash.
3. Now we will check if char b/w forward slash & backdash are valid (a...z or 1...9).
4. check that if the char is a ':' or another '\' or '/', then not valid.
5. Similarly check the ~~char~~ before '&/' using lower bound on sets of forward & back slash and will find no. of char b/w them.

Eg $n = 6,\ k = 3,\ a = 1$

No. of guesses : 1 2 4

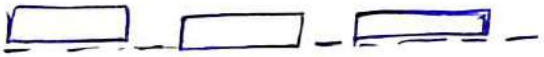
3 boxes each of size 1

[1] [2] [3]



Given, n, a → Max $k = \frac{n+1}{a+1}$

$n = 12,\ a = 3$



[1 ---- n] → max

↙ x ↘

[1---x-1] [x+1 --- n]

a + b ≥ k → When it fill the count is smaller than k, we will return that index.

```cpp
void solve()
{
    set<pair<int, int>>s;
    s.insert({1, 20});
    for (int i = 0; i < 5; i++)
    {
        int x;
        cin >> x;
        cout << "want to block: " << x << endl;
        auto it = s.lower_bound({x, 0});
        pair<int, int>p;
        if (it != s.end())p = *it;
        else
        {
            it--;
            p = *it;
        }
        if (p.first == x)
        {
            cout << "removing " << p.first << " " << p.second << endl;
            s.erase(it);
            int sz=p.second-p.first+1;
            if (p.first != p.second)
            {
                if (x + 1 <= p.second)
                    {
                        s.insert({x + 1, p.second});
                    }
            }
        }
        else if (p.first < x)
        {
            cout << "removing " << p.first << " " << p.second << endl;
            s.erase(it);
            cout << "inserting" << p.first << " " << x - 1 << " and " <<
```

```
                {
                    s.insert({x + 1, p.second});
                }
            }
        }
        else if (p.first < x)
        {
            cout << "removing " << p.first << " " << p.second << endl;
            s.erase(it);
            cout << "inserting" << p.first << " " << x - 1 << " and " <<
            if (x - 1 >= p.first)
                s.insert({p.first, x - 1});
            if (x + 1 <= p.second)
                s.insert({x + 1, p.second});
        }
        else
        {
            it--;
            p = *it;
            cout << "removing " << p.first << " " << p.second << endl;
            s.erase(it);
            cout << "inserting" << p.first << " " << x - 1 << " and " <<
            if (x - 1 >= p.first)
                s.insert({p.first, x - 1});
            if (x + 1 <= p.second)
                s.insert({x + 1, p.second});
        }
        cout << "printing set:\n";
        for (auto j : s)
        {
            cout << j.first << " " << j.second << endl;
        }
    }
}
signed main()
```