Abbreviation in C++

```
#include <iostream>
#include <string>
using namespace std;

class Abbreviation {
public:
    static void solution(string str, string asf, int count,
int pos) {
        if (pos == str.length()) {
            if (count == 0) {
                cout << asf << endl;
            } else {
                cout << asf << count << endl;
            }
            return;
        }

        if (count > 0) {
            solution(str, asf + to_string(count) + str[pos],
0, pos + 1);
        } else {
            solution(str, asf + str[pos], 0, pos + 1);
        }

        solution(str, asf, count + 1, pos + 1);
    }
};

int main() {
    string str = "pep";
    Abbreviation::solution(str, "", 0, 0);
    return 0;
}
```

Dry Run Table (Step-by-Step)

We'll list:

- pos: current position in the string
- count: how many characters we've skipped (abbreviated)
- asf: abbreviation-so-far

pos	char	count	asf	Recursive Call
0	p	0	""	choose 'p' → asf = "p"
1	e	0	"p"	choose 'e' → asf = "pe"
2	p	0	"pe"	choose 'p' → asf = "pep"
3	—	0	"pep"	**output: pep**
2	p	1	"pe"	skip 'p' (count = 1)
3	—	1	"pe"	**output: pe1**
1	e	1	"p"	skip 'e' (count = 1)
2	p	0	"p1p"	count > 0 → add 1 then 'p'
3	—	0	"p1p"	**output: p1p**
2	p	2	"p"	skip 'p' (count = 2)
3	—	2	"p"	**output: p2**
0	p	1	""	skip 'p' (count = 1)
1	e	0	"1e"	count > 0 → add 1, then 'e'
2	p	0	"1ep"	choose 'p'
3	—	0	"1ep"	**output: 1ep**
2	p	1	"1e"	skip 'p' (count = 1)
3	—	1	"1e"	**output: 1e1**
1	e	1	""	skip 'e'
2	p	0	"2p"	count = 2 → asf = "2p"
3	—	0	"2p"	**output: 2p**
2	p	2	""	skip 'p'
3	—	3	""	**output: 3**

✅ **Final Output:**

```
pep
pe1
p1p
```

	p2 1ep 1e1 2p 3
Output:- pep pe1 p1p p2 1ep 1e1 2p 3	