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;
}
```

Step-by-Step Execution:

The function solution() uses recursion to generate all the abbreviations. It has two main actions:

1. **Include the current character with an abbreviation count if any.**
2. **Include the current character as is.**

- str: The string for which we need to find abbreviations.
- asf: The abbreviation formed so far.
- count: The number of characters skipped (abbreviated).
- pos: The current position in the string.

1. **Initial Call**: solution("pep", "", 0, 0)
    - **pos = 0**, **count = 0**, **asf = ""**
    - Two options:
        1. Skip character at pos = 0 (first 'p')
        2. Include character at pos = 0 (first 'p')
    - Recur on both options.

2. **First Option**: Skip character at pos = 0
    - Call solution("pep", "", 1, 1) (increment count by 1)

3. **Call**: solution("pep", "", 1, 1)
    - **pos = 1**, **count = 1**, **asf = ""**
    - Two options:
        1. Skip character at pos = 1 (character 'e')
        2. Include character at pos = 1 (character 'e')
    - Recur on both options.

4. **First Option**: Skip character at pos = 1
    - Call solution("pep", "1", 2, 2) (skip 'e' and increment count)

5. **Call**: solution("pep", "1", 2, 2)
    - **pos = 2**, **count = 2**, **asf = "1"**
    - Two options:
        1. Skip character at pos = 2 (second 'p')
        2. Include character at pos = 2

(second 'p')
- Recur on both options.

6. **First Option**: Skip character at pos = 2
   - Call solution("pep", "12", 3, 3) (skip 'p')

7. **Call**: solution("pep", "12", 3, 3)
   - **pos = 3**, **count = 3**, **asf = "12"**
   - Base case reached (pos == str.length())
   - Output: "12"

8. **Second Option**: Include character at pos = 2
   - Call solution("pep", "1p", 0, 3) (include 'p' and reset count)

9. **Call**: solution("pep", "1p", 0, 3)
   - **pos = 3**, **count = 0**, **asf = "1p"**
   - Base case reached (pos == str.length())
   - Output: "1p"

10. **Backtrack to Call 4**: solution("pep", "", 1, 1)
    - **pos = 1**, **count = 1**, **asf = ""**
    - Second option: Include character at pos = 1 ('e')
    - Call solution("pep", "e", 0, 2) (reset count and include 'e')

11. **Call**: solution("pep", "e", 0, 2)
    - **pos = 2**, **count = 0**, **asf = "e"**
    - Two options:
      1. Skip character at pos = 2 (second 'p')
      2. Include character at pos = 2 (second 'p')
    - Recur on both options.

12. **First Option**: Skip character at pos = 2
    - Call solution("pep", "e1", 1, 3) (skip

'p' and increment count)

13. **Call**: solution("pep", "e1", 1, 3)
    - **pos = 3**, **count = 1**, **asf = "e1"**
    - Base case reached (pos == str.length())
    - Output: "e1"

14. **Second Option**: Include character at pos = 2
    - Call solution("pep", "ep", 0, 3) (include 'p' and reset count)

15. **Call**: solution("pep", "ep", 0, 3)
    - **pos = 3**, **count = 0**, **asf = "ep"**
    - Base case reached (pos == str.length())
    - Output: "ep"

16. **Backtrack to Initial Call**: solution("pep", "", 0, 0)
    - **pos = 0**, **count = 0**, **asf = ""**
    - Second option: Include character at pos = 0 ('p')
    - Call solution("pep", "p", 0, 1) (include 'p' and reset count)

17. **Call**: solution("pep", "p", 0, 1)
    - **pos = 1**, **count = 0**, **asf = "p"**
    - Two options:
        1. Skip character at pos = 1 (character 'e')
        2. Include character at pos = 1 (character 'e')
    - Recur on both options.

18. **First Option**: Skip character at pos = 1
    - Call solution("pep", "p1", 1, 2) (skip 'e' and increment count)

19. **Call**: solution("pep", "p1", 1, 2)
    - **pos = 2**, **count = 1**, **asf = "p1"**
    - Two options:
        1. Skip character at pos = 2

(second 'p')
2. Include character at pos = 2 (second 'p')
- Recur on both options.

20. **First Option**: Skip character at pos = 2
    - Call solution("pep", "p12", 2, 3) (skip 'p' and increment count)

21. **Call**: solution("pep", "p12", 2, 3)
    - **pos = 3**, **count = 2**, **asf = "p12"**
    - Base case reached (pos == str.length())
    - Output: "p12"

22. **Second Option**: Include character at pos = 2
    - Call solution("pep", "p1p", 0, 3) (include 'p' and reset count)

23. **Call**: solution("pep", "p1p", 0, 3)
    - **pos = 3**, **count = 0**, **asf = "p1p"**
    - Base case reached (pos == str.length())
    - Output: "p1p"

24. **Backtrack to Call 17**: solution("pep", "p", 0, 1)
    - **pos = 1**, **count = 0**, **asf = "p"**
    - Second option: Include character at pos = 1 ('e')
    - Call solution("pep", "pe", 0, 2) (include 'e' and reset count)

25. **Call**: solution("pep", "pe", 0, 2)
    - **pos = 2**, **count = 0**, **asf = "pe"**
    - Two options:
        1. Skip character at pos = 2 (second 'p')
        2. Include character at pos = 2 (second 'p')
    - Recur on both options.

26. **First Option**: Skip character at pos = 2

	○ Call solution("pep", "pe1", 1, 3) (skip 'p' and increment count) 27. **Call**: solution("pep", "pe1", 1, 3) ○ **pos = 3**, **count = 1**, **asf = "pe1"** ○ Base case reached (pos == str.length()) ○ Output: "pe1" 28. **Second Option**: Include character at pos = 2 ○ Call solution("pep", "pep", 0, 3) (include 'p' and reset count) 29. **Call**: solution("pep", "pep", 0, 3) ○ **pos = 3**, **count = 0**, **asf = "pep"** ○ Base case reached (pos == str.length()) ○ Output: "pep"
Output:- pep pe1 p1p p2 1ep 1e1 2p 3	