Subsequence in C++

```
#include <iostream>
#include <string>

using namespace std;

void sol(string q, string a) {
    if (q.length() == 0) {
        cout << a << "-" << endl;
        return;
    }

    char ch = q[0];
    string rest = q.substr(1);
    sol(rest, a);
    sol(rest, a + ch);
}

int main() {
    string s = "abc";
    sol(s, "");

    return 0;
}
```

Initial Setup:

- Call sol("abc", "") to generate all subsequences of the string.

Step-by-Step Execution:

1. **First Call: sol("abc", "")**
    - **q = "abc", a = ""**.
    - Take the first character 'a' from the string and split it into:
        - **ch = 'a'**, **rest = "bc"**.
    - **Recursively call** sol("bc", "") to handle the case where 'a' is **not** included.
    - **Recursively call** sol("bc", "a") to handle the case where 'a' is **included**.

2. **Second Call: sol("bc", "")**
    - **q = "bc", a = ""**.
    - Take the first character 'b' from the string and split it into:
        - **ch = 'b'**, **rest = "c"**.
    - **Recursively call** sol("c", "") to handle the case where 'b' is **not** included.
    - **Recursively call** sol("c", "b") to handle the case where 'b' is **included**.

3. **Third Call: sol("c", "")**
    - **q = "c", a = ""**.
    - Take the first character 'c' from the string and split it into:
        - **ch = 'c'**, **rest = ""**.
    - **Recursively call** sol("", "") to handle the case where 'c' is **not** included.
    - **Recursively call** sol("", "c") to handle the case where 'c' is **included**.

4. **Base Case: sol("", "")**
    - **q is empty**, print the current subsequence: "-".

5. **Base Case: sol("", "c")**
    - **q is empty**, print the current subsequence: "c-".

6. **Return to Third Call: sol("c", "b")**
    - **Recursively call** sol("", "bc") to handle the case where 'c' is **included**.

7. **Base Case: sol("", "bc")**
    - **q is empty**, print the current subsequence: "bc-".

8. **Return to Second Call: sol("bc", "")**
    - Now handle the case where 'b' is **included**:
        - **sol("c", "b")** has been handled, now process the second part.

Continuation for the First Character 'a':

1. **Second Part: sol("bc", "a")**
    - **q = "bc", a = "a"**.
    - Take the first character 'b' from the string and split it into:
        - **ch = 'b'**, **rest = "c"**.
    - **Recursively call** sol("c", "a") to handle the case where 'b' is **not** included.
    - **Recursively call** sol("c", "ab") to handle the case where 'b' is **included**.

2. **Third Call: sol("c", "a")**
    - **q = "c", a = "a"**.
    - Take the first character 'c' from the string and split it into:
        - **ch = 'c'**, **rest = ""**.
    - **Recursively call** sol("", "a") to handle the case where 'c' is **not** included.
    - **Recursively call** sol("", "ac") to handle the case where 'c' is **included**.

3. **Base Case: sol("", "a")**
    - **q is empty**, print the current subsequence: "a-".

4. **Base Case: sol("", "ac")**
    - **q is empty**, print the current subsequence: "ac-".

Final Part of the Execution:

1. **Return to Second Call: sol("c", "ab")**
    - **Recursively call** sol("", "abc") to handle the case where 'c' is **included**.

2. **Base Case: sol("", "abc")**
    - **q is empty**, print the current

	subsequence: "abc-".

Output:-
-
c-
b-
bc-
a-
ac-
ab-
abc-