Lab A: Anagrams
Consider the following class:

```cpp
#include <cstring>

class Anagram {
    char s[100];

public:
    Anagram() {...}
    Anagram(const char * t) {...}

    int length() {...}
    bool isAnagram(const char * t) {...}
    void generate(char * t, int i) {...}
};
```

(a) Implement the default constructor to make `s` an empty string.

(b) Implement the other constructor to copy `t` into `s` while ignoring spaces in `t`, and handle the case when `t` is too long. Assume `t` is null terminated.

(c) Implement the length function to return the length of `s` (not 100).

(d) Implement the `isAnagram` function to return true if and only if `t` is a permutation of `s` when spaces in `t` are ignored. Assume `t` is null terminated. Your function should not change `s` or `t`, and should not make copies of them. In other words, the amount of additional memory used by your function should be independent of the lengths of `s` and `t`.

(e) [if there is time] Implement the `generate` function to make `t` a random permutation of `s` plus `i` spaces. The spaces must not be at the beginning or end of `t` and none of them are consecutive. Assume `0 ≤ i < strlen(s)` and `t` is long enough, i.e. `strlen(t) ≥ strlen(s) + i`. 