CSCI 135 Software Design and Analysis, C++
Lab 10

Saad Mneimneh
Hunter College of CUNY

Lab A: Anagrams
Consider the following class:

```
#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 non of them are consecutive. Assume $0 \leq i <$ strlen(s) and t is long enough, i.e. strlen(t) $\geq$ strlen(s) $+ i$.

Solution:

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstring>

using std::cout;

void swap(char * t, int i, int j) {
  char temp=t[i];
  t[i]=t[j];
  t[j]=temp;
}

class Anagram {
  char s[100];
 public:
  Anagram();
  Anagram(const char * t);

  int length();
  bool isAnagram(const char * t);
  void generate(char * t, int k);
};

//we are going to look at this style
//of implementing a class next time

Anagram::Anagram() {
  s[0]='\0';
}

Anagram::Anagram(const char * t) {
  int j=0;
  for (int i=0; t[i]!='\0'; i=i+1)
    if (t[i]!=' ') {
      s[j]=t[i];
      j=j+1;
    }
  s[j]='\0';
}

int Anagram::length() {
  return strlen(s);
}
```

```
bool Anagram::isAnagram(const char * t) {
  int s_len=strlen(s);
  int t_len=strlen(t);

  //number of non-space characters in t
  int l=0;
  for (int i=0; i<t_len; i=i+1)
    if (t[i]!=' ')
      l=l+1;

  if (s_len!=l) //t cannot be an anagram
    return false;

  //now let's do the work
  //make sure every character in s appears
  //the same number of times in t
  int count1;
  int count2;

  for (int i=0; i<s_len; i=i+1) {
    //count in s
    count1=0;
    for (int j=0; j<s_len; j=j+1)
      if (s[i]==s[j])
        count1=count1+1;

    //count in t
    count2=0;
    for (int j=0; j<t_len; j=j+1)
      if (s[i]==t[j])
        count2=count2+1;

    if (count1!=count2)
      return false;
  }

  //if here, it must be true
  return true;
}
```

```
void Anagram::generate(char * t, int k) {
  //this assumes 0<=k<strlen(s)

  //permuting s does not hurt
  int s_len=strlen(s);
  for (int i=s_len-1; i>0; i=i-1)
    swap(s, i, rand()%(i+1));

  //the first k+1 elements will be the start of k+1
  //groups of characters, separated by k spaces

  int index=0;
  for (int i=0; i<=k; i=i+1) {
    t[index]=s[i]; //the opening of the ith group
    index=index+1;
    for (int j=k+1; j<s_len; j=j+1)
      if (i+rand()%(k+1-i)==i) { //falls into the group
        t[index]=s[j];
        index=index+1;
        swap(s, j, s_len-1);  //move this element to the end
        s_len=s_len-1;        //and ignore it from now on
      }
    t[index]=' ';
    index=index+1;
  }
  t[index-1]='\0';
}

int main() {
  srand(time(0));
  Anagram a=Anagram("something you like");
  char t[100];
  cout<<a.isAnagram("you like something")<<'\n'
  a.generate(t,3);
  cout<<t<<'\n';
}
```