Assignment:

Write a function num_dups that takes in a string and returns the number of duplicated characters in the string.

Hint: You can use a std::map to maintain a count of the characters.

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

int num_dups(string str) {
//code
}
```

Solution:

I've made two versions of this solution. First one implies that input "Hello" provides the result equals "1" (it means that letter "l" duplicates only once):

```
int num_dups(string str) {
	string copy_str = str;
	int num_of_dups = 0;
	for (int symbol = 0; copy_str[symbol] != '\0'; symbol++)
	{
		for (int comp_symbol = symbol + 1; copy_str[comp_symbol] != '\0'; comp_symbol++)
		{
			if (copy_str[symbol] == copy_str[comp_symbol] && ( (copy_str[comp_symbol] >= 'A' &&
copy_str[comp_symbol] <= 'Z') || (copy_str[comp_symbol] >= 'a' && copy_str[comp_symbol] <= 'z') ))
			{
				rep = true;
				num_of_dups++;
				copy_str[comp_symbol] = '#';
			}
		}
	}
	return num_of_dups;
}
```

The second one implies that input "Hello" provides the result equals "2" (it means that letter "l" duplicates only once, but function also counts the first entry of this letter):

```
int num_dups(string str) {
	string copy_str = str;
	int num_of_dups = 0;
	for (int symbol = 0; copy_str[symbol] != '\0'; symbol++)
	{
		bool rep = false;
		for (int comp_symbol = symbol + 1; copy_str[comp_symbol] != '\0'; comp_symbol++)
		{
			if (copy_str[symbol] == copy_str[comp_symbol] && ( (copy_str[comp_symbol] >= 'A' &&
copy_str[comp_symbol] <= 'Z') || (copy_str[comp_symbol] >= 'a' && copy_str[comp_symbol] <= 'z') ))
			{
				rep = true;
				num_of_dups++;
				copy_str[comp_symbol] = '#';
			}
		}
		if (rep)
		{
			num_of_dups++;
		}
	}
	return num_of_dups;
}
```