Two different types of Strings: string objects and C-Strings

In the C++ language, there are two primary ways that strings are stored in memory: as string objects or as C-strings.

1. Using string objects

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

int main() {
  string line;
  cout << "Enter a sentence? ";
  getline(cin, line);

  cout << line;
}
```

String operators you should be familiar with:

• =	- string assignment
• ==, <, >	- string comparisons
• +	- string concatenation
• [ ]	- character at indexed position of string

String functions you should be familiar with:

• length()	- returns the length of a string
• find(str, x)	- returns the index at or beyond x where str is found in a string
• replace(start, end, new)	
• substr(x, n)	- returns a substring from a string starting at index x for n characters

Here’s a sample program using these functions.

```
#include <iostream>
#include <string>
using namespace std;
// Reference: http://www.cplusplus.com/reference/string/string/find/

int main() {
  string str = "There are two needles in this haystack with needles.";
  string str2 = "needle";

  cout << "The sentence" << endl;
  cout << str << endl;
  cout << "has " << str.length() << " characters" << endl;

  // find first needle in the sentence
  int found = str.find(str2);
  // If no matches were found, the function returns string::npos.
  if (found != string::npos)
  // if (found > 0)
    cout << "first 'needle' is found at: " << found << '\n';

  // find second needle in the sentence
  found=str.find(str2, found+1);
  if (found > 0)
    cout << "second 'needle' is found at: " << found << '\n';

  // replace the first needle with rat:
  cout << "Here's the sentence after replacing needle with rat" << endl;
  str.replace(str.find(str2), str2.length(), "rat");
  cout << str << '\n';

  // find the 4th word in the sentence
  found = 0;
  for (int i=0; i<3; i++) {
    found = str.find(" ", found+1);
    cout << found << endl;
  }
  // extract the 4th word in the sentence
  int length = str.find(" ", found+1) - found - 1; // find length of word
  str2 = str.substr(found+1, length);
  cout << "The fourth word is \"" << str2 << "\"." << endl;

  // extract the 4th character in the sentence
  cout << "The fourth character is \"" << str[3] << "\"." << endl;

  // looping through the entire sentence
  cout << "Here's the sentence again printed out one character at a time" <<
endl;
  int i=0;
  while (str[i] != '\0') {
    cout << str[i];
    i++;
  }
  cout << endl;

  return 0;
}
```

And here's the output from the program

```
The sentence
There are two needles in this haystack with needles.
has 52 characters
first 'needle' is found at: 14
second 'needle' is found at: 44
Here's the sentence after replacing needle with rat
There are two rats in this haystack with needles.
5
9
13
The fourth word is "rats".
The fourth character is "r".
Here's the sentence again printed out one character at a time
There are two rats in this haystack with needles.
```

2. *Using C-strings*

A C-string is simply a char array that contains a string of characters terminated by the null character '\0'. The null character has the ASCII value of 0.

```
#include <iostream>
#include <cstring>  // needed for strlen(), strcmp(), strcpy()
#include <cctype>   // needed for isalpha(), isalnum(), etc.
using namespace std;

int main() {
  const int SIZE = 80;
  char line[SIZE];
  cout << "Enter a sentence? ";
  cin.getline(line, SIZE);

  int i=0;
  while (line[i] != '\0') {
    cout << line[i];    // print out one character at a time
    i++;
  }
  cout << endl << "There are " << i << " characters in the sentence.";
}
```

C-string functions you should be familiar with:

- isalpha, isalnum, isdigit, islower, ispunct, isupper, isspace
  - returns true if character is one of the specified character for that function
- toupper, tolower
  - returns the uppercase or lowercase of the alphabet
- strlen(cstr) - returns the length of a c-string
- strcmp(cstr1, cstr2) - compares two C-strings. Returns 0 if they are equal. Returns a positive number if cstr1 is greater than cstr2.
- strcpy(cstr_destination, cstr_source)
  - copies the source C-string to the destination C-string
- atoi("123") - converts a C-string to an integer
- atof("3.14") - converts a C-string to a float

Here's a sample program using these functions.

```
#include <iostream>
#include <cstring> // needed for strlen()
#include <cctype> // needed for isalpha(), etc.
using namespace std;

int main() {
    const int SIZE = 80;
    char line[SIZE];
    cout << "Enter a sentence? ";
    cin.getline(line, SIZE);
    int i = 0;
    while (line[i] != '\0') {
        cout << line[i]; // print out one character at a time
        i++;
    }
    cout << endl << "There are " << i << " characters in the sentence using manual
count." << endl;

    cout << "There are " << strlen(line) << " characters in the sentence using
strlen." << endl;

    char word1[SIZE], word2[SIZE];
    cout << endl;
    cout << "Enter a word? ";
    cin.getline(word1, SIZE);
    cout << "Enter another word? ";
    cin.getline(word2, SIZE);

    int result = strcmp(word1, word2);
    if (result == 0) {
        cout << "The two words are the same" << endl;
    }
    else if (result > 0) {
        cout << word2 << " comes before " << word1 << endl;
    }
    else {
        cout << word1 << " comes before " << word2 << endl;
    }

    char number[SIZE];
    cout << endl;
    cout << "Enter a number? ";
    cin.getline(number, SIZE);
    cout << "Doubling the number is " << atof(number) * 2 << endl;
}
```

Exercises using c-strings (Problems with an asterisk are more difficult)

1. Write a program to input a c-string, and then print out the same c-string but with all of the letters capitalized.

2. Write a program to input a c-string, and then use a loop to count to find out the length of the string, i.e. without using the strlen() function. Compare your count with the number returned by the strlen() function to see if they are the same. Print out appropriate messages.

3. Write a program to input a c-string, and then print out how many lower case, how many upper case, and how many printable non-alphanumeric characters are in the c-string.

4. Write a program to input a sentence using a c-string. Print out how many words are in the sentence. Assume that all the words are separated by exactly one space. You need to use a loop to scan through the entire c-string array.

5. * Just like question 4, but there can be two or more consecutive spaces separating words.

6. Write a program to input a sentence using a c-string. Print out the sentence with the first letter of every word capitalized. You need to use a loop to scan through the entire c-string array.

Exercises using string objects (Problems with an asterisk are more difficult)

7. Write a program to input a sentence using a string. Print out how many words are in the sentence. Instead of scanning through the characters one by one in the string, you need to use the find() string library function to search for the spaces.

8. Write a program to input a sentence using a string. Print out the sentence with the first letter of every word capitalized. You need to use the replace() string library function to replace the first letter of every word.

9. Write a program to first input a sentence using the string type. Then input a word. Print out a message saying whether the word is in the sentence or not. You need to use of some of the string library functions.

10. ** Another guessing game. Create a pattern string of random length from 5 up to 10 characters long, and initialize it with random digits, e.g. 3687073452. The user will then try to guess the digits one at a time. The pattern string is hidden from the user, and the program just prints out *'s to represent this pattern string., e.g. **********

    The user will input a digit. If the digit is found in the pattern, then the location of where that digit is at is revealed by changing the * at that location with the digit. For example, if the user enters a 6 then the program will print *6********. If the digit is not found in the pattern then do nothing.

    If there are more than one occurrences of that digit then reveal the first one. For example, if the user enters a 7 then the program will print *6*7******. If the user enters another 7 then the program will print *6*7*7****

    Keep repeating until the user has guessed all of the digits

11. ** Password checking. Write a program for the user to enter a password and then check to see if the password is valid or not. A valid password must satisfy all of the following criteria:

    - at least 10 characters long
    - contains at least one digit
    - contains at least one lower-case letter
    - contains at least one upper-case letter
    - contains at least one non-alphanumeric character
    - does not contain any consecutive duplicate characters (e.g. `"aa"`)

    A prompt with the criteria listed is first printed, followed by the entering of the password. After entering the password, it is then checked to see whether it satisfies all of the above criteria. If it does then print out the message "**Your password has been accepted.**" Otherwise, print out the message "**Your password does not meet the above criteria.**"

    The following is a sample run of the program.

```
Please enter a password with the following criteria:
  - at least 10 characters long
  - contains at least one digit
  - contains at least one lower-case letter
  - contains at least one upper-case letter
  - contains at least one non-alphanumeric character
  - does not contain any consecutive duplicate characters (e.g. "aa")

Password: Padingtons1*
Your password has been accepted.
```

```
Please enter a password with the following criteria:
  - at least 10 characters long
  - contains at least one digit
  - contains at least one lower-case letter
  - contains at least one upper-case letter
  - contains at least one non-alphanumeric character
  - does not contain any consecutive duplicate characters (e.g. "aa")

Password: Paddingtons
Your password does not meet the above criteria.
```

12. ** Password checking. Continuing from question 11 above, when the password does not meet the criteria, the program will also print out which criteria are not satisfied.

13. **** Write a program to count the words in the following passage. It will output a list of words and the number of times they appear in the passage. Ignore all punctuation characters like commas and periods. Treat uppercase and lowercase as different characters. You can, if you like, treat the word “farmer’s” as two words to make it simpler.

    The program will print out the words and their counts like the following

    ```
    Dorothy - 2
    lived - 1
    in - 7
    the - 11
    etc.
    ```

    The passage is hard-coded into the program by initializing a string constant with it, like this

    ```
    const string passage = "Dorothy lived in the midst of the great Kansas
      prairies, with Uncle Henry, who was a farmer, and Aunt Em, who was the
      farmer’s wife. Their house was small, for the lumber to build it had to be
      carried by wagon many miles. There were four walls, a floor and a roof,
      which made one room; and this room contained a rusty looking cookstove, a
      cupboard for the dishes, a table, three or four chairs, and the beds. Uncle
      Henry and Aunt Em had a big bed in one corner, and Dorothy a little bed in
      another corner. There was no garret at all, and no cellar except a small
      hole dug in the ground, called a cyclone cellar, where the family could go
      in case one of those great whirlwinds arose, mighty enough to crush any
      building in its path. It was reached by a trap door in the middle of the
      floor, from which a ladder led down into the small, dark hole.";
    ```

    Hint: one way to solve this problem is to use two parallel arrays; the first array is of type string to store the words and the second array is of type int to store the count for the corresponding word at the same index. For example, the string array at index 0 contains the word “Dorothy” and the corresponding int array at index 0 will have the count of 2 at the end. You need to parse the passage for each word. After getting a word, you need to search through the string array for the word. If it is found then increment the count in the int array using the same index for where you found the word in the string array. If it is not found then insert the word into the string array and set the count at the same index in the int array to one. You'll also need to keep track of how many words you already have added into the string array (which will be the same as the count array) so that you know where to add the new word.

    Note that your program should work for any given passage and not only for this one. The program is automatically updating the string array with each new word found and updating the word count in the int array. You are NOT initializing the arrays manually yourself with the words and counts.