CS 185
Programming Assignment 6

Before writing any code, you should write down the steps (in English) you need to take to solve these problems. This is your *pseudocode*. Once you have your pseudocode written, you can convert it into C++ code. If you can't write down the steps in English, you certainly can't write it in C++.
Note: You will not submit the pseudocode!!!

Details

All of the functions you'll be writing today deal with NULL-terminated C-style strings. The only header file you can include is `string.h`, and you'll probably only need the `strlen` function.

1. The first function you will write will convert a string of digits into an integer. It is sort of like the C / C++ standard library function `atoi`. The name of the function is `strtoint` (string to int) and the prototype is:

   ```
   int strtoint(const char string[]);
   ```

   You will be given a valid NULL-terminated C-style string as the only parameter and will return the integer value of the string. The string will be a valid integer representation and may be negative. Some valid strings are:

   ```
   "0"
   "1"
   "123"
   "1234567890"
   "-1"
   "-0"
   "-1234567890"
   ```

   There will be no leading + (plus sign) on positive integers and no spaces in the strings.
   You must not use the `pow()` function. You may write your own `power` function, if you need one, but it's not required.

   Hint: Work on getting positive numbers working and then try to get the negative numbers working. Like all problems, there are several ways that you can solve this. One way is to recognize the pattern in base 10 numbers:

   $$1234 = 4 + 30 + 200 + 1000$$
   $$= (4 * 1) + (3 * 10) + (2 * 100) + (1 * 1000)$$
   $$= (4 * 10^0) + (3 * 10^1) + (2 * 10^2) + (1 * 10^3)$$

Using pointers or subscripts, you can extract the '4', '3', '2', and '1' characters, convert them to digits (e.g. `'4' - '0' == 4`), and then multiply by the correct power of ten. Another way is to work left to right, extracting the current character and multiplying by 10 and adding the next character, multiplying by ten, adding the next, etc. like this:

```
1234 = ((((1 * 10) + 2) * 10) + 3) * 10) + 4
```

You'll have to do some slight additional work to handle negative numbers.

2. The second function works the opposite way: converts an integer into a string. The name of the function is `inttostr` (int to string) and the prototype is:

```
void inttostr(int number, char string[]);
```

The string you are given is large enough to contain the largest integer (10 digits), including the negative sign and NULL terminator.

Hint: Work on the positive numbers first before trying to deal with the negative numbers. You will use the mod operator, `%`, and the division operator, `/`, in a loop to extract the digits from the integer. Then convert each integer to a character (e.g. `4 + '0' == '4'` and put the character in the proper position in the string. It may be easier to build the string backwards (in reverse), and when you've finished building the string, just reverse it in place. (You already know how to reverse an array, as you did it in a previous homework assignment.)

3. The third function will reverse the words in a string. For example:

```
Four score and seven years ago
```

becomes

```
ago years seven and score Four
```

Notice that you are not reversing the characters in the words, just the words. The function prototype is:

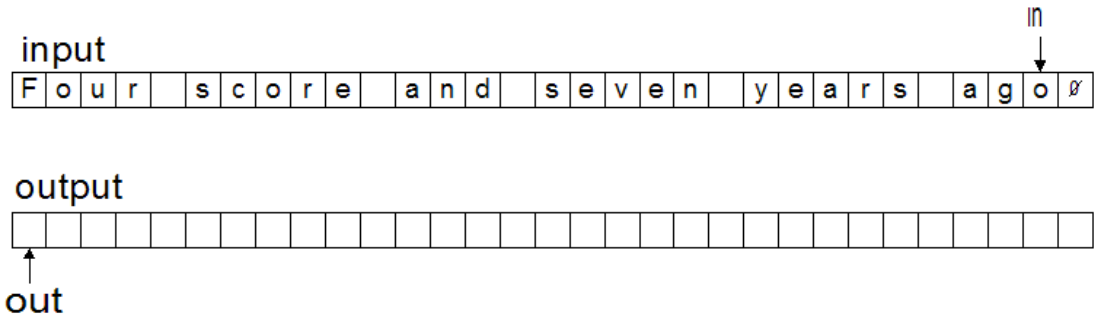
```
void reverse_words1(const char input[], char output[]);
```

The input is a valid NULL-terminated C-style string and the output the same size as the input, so you will have enough room for all of the characters including the NULL terminator.

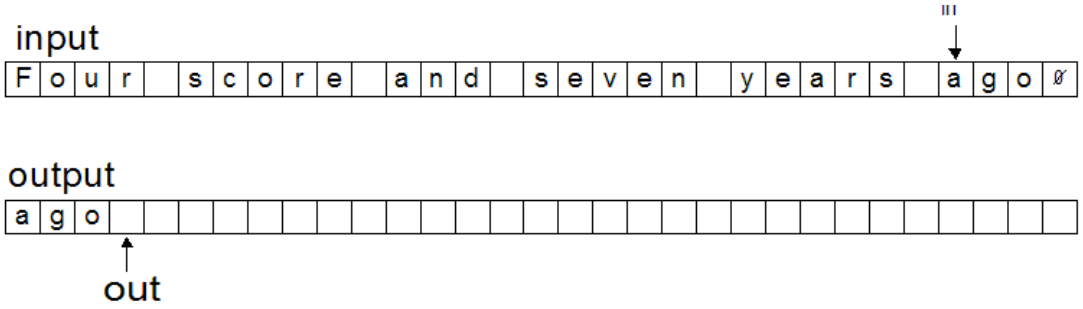
Hint: Walk the input string backwards looking for a space character. Then copy each character of the word to its right into the output. Do this until you get to the beginning of the string and copy the first word as the last word of the output.

Here are some diagrams to help get you started:

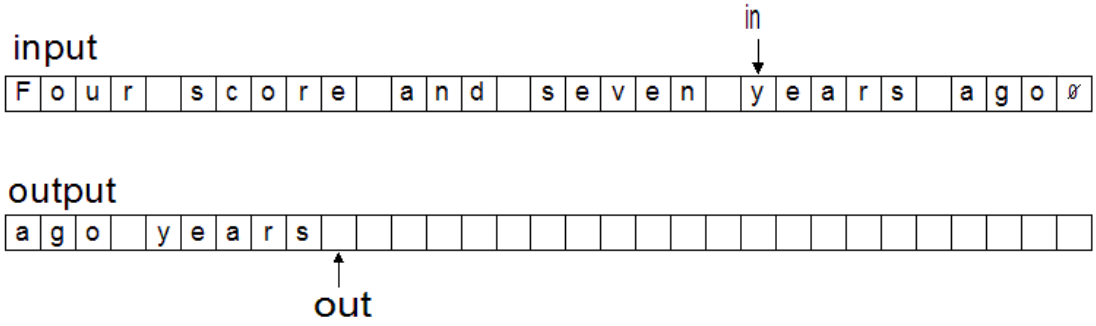
At the start:



Found the last word (ago) in **input**, then copied it to **output**:



Found the word "years" in the input, then copied it to the output:



4. The fourth function is very similar to the third function except that you will reverse the words in place. This means there is no extra memory required. The prototype looks like this:

```
void reverse_words2(char input[]);
```

There's a clever trick that will make this easier than the first reverse function. First, reverse the entire string character-by-character:

```
This is a string
```

becomes

```
gnirts a si sihT
```

You'll notice that all of the words are in their proper position. They're just reversed. Now, simply reverse the letters of each word. To make this simpler, you should create a helper function that reverses all of the characters between two pointers:

```
/* Helper function to reverse a range a characters */
void reverse(char *start, char *end)
{
  /* Reverse the characters from start to end */
}
```

You can even use this function to reverse the entire string at the start.

You are given a file called **main.cpp**, which includes the main function with several test cases for you to use. There are four functions prototyped in that file, one for each of the functions you are to write. As always, you must implement both of these functions *exactly as prototyped.* You will implement these functions in a file called **reverse.cpp**.

NOTE:
Notice the use of the `#if 0 ... #endif` in the file. This is a simple way to enable/disable a bunch of code in the file. You shouldn't try to pass every test at once. Work on the first test, which is very simple. When you get that test working, move the `#if 0` around the next one and see if you pass that test. Continue in this way until you pass all of the tests. All the header files that you need are already included.

An **output.txt** file is given to you so that you can compare your output with the expected output.

Note: ***I expect the exact output. So be careful and use a tool (such as: WinMerge) that will check the difference between two text files.***

This assignment will not require you to include any header files in your code. All of the code that you write must be in the two files mentioned above since you will not be turning in ***main.cpp***.

Comments

In this and future assignments, you are required to include:

- **A file header comment in every piece of source file. The format is shown in the "Comments.cpp" file given to you in the beginning of the semester and should be present at the very top of all your code.**
- **Function header for each function you create. The format is shown in the "Comments.cpp" file given to you in the beginning of the semester and should be present at the top of every function.**
- **Inline commenting for your code.**

What to submit

You must submit the CPP files (***reverse.cpp***) in a single .zip file (go to the class page on moodle and you will find the assignment submit link). ***Do not submit any other files than the ones listed.***

If you've forgotten how to submit files, the details are posted in the syllabus and in the assignment guidelines document. Failure to follow the instructions will result in a poor score on the assignment (and possibly a zero).

Special note:

The due date/time posted is the positively latest you are allowed to submit your code. Since the assignments can easily be completed well before the deadline, you should strive to turn it in as early as possible. If you wait until the deadline, and you encounter unforeseen circumstances (like being sick, or your car breaking down, or something else), you may not have any way to submit the assignment on time. Moral: **Don't wait until the last day to do your homework.**