Assignment 6

In this assignment, you will move code from previous homework and programming examples and place it into a pair of files to create a C **module**. The module lets us separate code, organize and isolate it from other modules. The idea is similar to how we build classes in higher level languages, and the python module idea comes from C.

You can get a B by successfuly completing Problem 1. Complete both successfully for an A grade.

Problem 1

Create a module called `mystring` from the code provided in the example `simple_word_count.c`.

Step 0 - make a new directory for your project called `assignment6`.

Copy the file `simple_word_count.c` into the directory.

Step 1 - make a new file called `mystring.c`

Open this file and add statements to include the system modules you need. To do this, copy the `#include` statements from `simple_word_count.c`.

Next move put the following functions into this file:

```
int stringlength(char* s)
{
...
}

int stringcompare(char* left, char* right)
{
...
}

void stringcopy(char* p1, char* p2)
{
...
}

char* makestring(char* buffer)
{
...
}

int getword(char* word, int lim, FILE* fp)
{
 ...
}
```

Step 2 - make a new header file called `mystring.h`

First, you need to put gaurds on the header file. It should look something like this:

```
#ifndef MYSTRING_H
#define MYSTRING_H

... code will go here...

#endif
```

The first two lines will be at the top of the file and the third line will be the last line in the file. These statements keep our program from defining the constants and functions in this header multiple times when they are included in multiple files.

Next, you will place all of the constants we had from our previous `simple_word_count.c` into this new file:

```
#define FILE_BF_SZ  1024
#define WORD_ARRAY_SZ 100
#define WORD_BF_SZ  100

#define TRUE 1
#define FALSE 0
```

Finally, add the protoypes of the functions in `mystring.c` into the file:

```
int stringlength(char* s);

int stringcompare(char* left, char* right);

void stringcopy(char* p1, char* p2);

char* makestring(char* buffer);

int getword(char* word, int lim, FILE* fp);
```

Step 3 - clean up `simple_word_count.c`

Remove the constants and functions from the file that were copied to the other files. only `int main()` remains.

add back the following two included modules at the top of the file:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

#include "mystring.h"

...
```

Finally, add the following line to `mystring.c`.

```
#include "mystring.h"
```

Step4 - Compile and run this program.

compile this program with the following command line:

```
gcc -o swc mystring.c simple_word_count.c
```

Fix any errors. When the compile works ok, run the program with:

```
./swc corpora/austen-emma.txt
```

to make sure the program runs correctly.

Problem 2

Using the work you did in assignment 5, create a new module with the String structure that includes the functions you wrote for memory management.

You can start by creating a directory called `StringStruct`.

Then, similar to the steps in Problem 1, create the files `StringStruct.c` and `StringStruct.h`. Place the `makeString(...)` program you wrote for Assignment 5 into `StringStruct.c`. Place the definition of the structure `String` into the file `StringStruct.h`.

Next, write versions of the functions:

```
unsigned int Stringlength(String* s);

// Stringcompare
// returns:
// 0 if equal
// 1 if left comes before right,
// -1 if right comes before left.
int Stringcompare(String* left, String* right);

String* Stringcopy(String* p1); // Note that makeString does most of the work for this one.

String* makeString(char* instring); // Assignment 5

void freeString(String* inString);
```

The standard module file `string.h` contains declarations for doing these functions with char* character array strings, so you can use the functions `strcpy`, `strlen`, and `strcmp` to help build the functions declared above for the String structure.

For `Stringcompare`, you can use the safer version of string comparison, `strncmp`, since the structure of the String object gives you information about the length of the string.

Include the definition of the functions (the code) in `StringStruct.c` and the declaration of the functions in `StringStruct.h`. See Problem 1 for the general pattern.

Write a `int main(...)`in a new file called `String_main.c`. Write tests for each of these 3 functions to see that they work. Here is a main function you can use for this:

```
// String_main.c - test string procedures.
//
#include <stdio.h>

#include "StringStruct.h"

int main(void)
{
  char* str1 = "Time flies like an arrow.  Fruit flies like a banana.";

  String* example = makeString(str1);

  // Check length
  printf("the example string is %u bytes long.\n", Stringlength(example));

  // copy the string
  String* copyofexample = Stringcopy(example);
  printf("example:\n%s\n\ncopy:\n%s\n", example->stringtext, copyofexample->stri

  // compare two strings
  String* s1 = makeString("string1");
  String* s2 = makeString("string2");
  printf("string compare of \n%s\nand\n%s\n is %d\n",
      s1->stringtext,
      s2->stringtext,
      Stringcompare(s1, s2));

}
```

To turn in your code, just zip the directories into a zip file and submit to canvas. Feel free to ask any questions as always.