Building the string object: phase5

Lab details:

We will continue working with our string object. This lab assumes that you completed all of the previous phases to build the string object.

Now, we are getting ready to finish all the functions needed by our string object.

Open your header file (string.h) and add the following declarations to your header file:

```
//Precondition: hResult and hAppend are handles to valid String objects.
//Postcondition:  hResult is the handle of a string that contains the original
//   hResult object followed by the hAppend object concatenated together. This
//   function should guarantee no change to the hAppend object and return
//   SUCCESS if they operation is successful and FAILURE if the hResult object
//   is unable to accomodate the characters in the hAppend string perhaps
//   because of a failed resize operation.  On FAILURE, no change to either
//   string should be made.
Status string_concat(String_Ptr hResult, String_Ptr hAppend);

//Precondition: hString is the handle to a valid String object.
//Postcondition: Returns an enumerated type with value TRUE if the string
//   is empty and FALSE otherwise.
Boolean string_empty(String_Ptr hString);
```

After you are done, run the following driver program on the whole string object to make sure that it is working correctly:

```
#include <stdlib.h>
#include <stdio.h>
#include "string.h"


//Simple.txt file includes the following sentence:
//The quick brown fox jumped over the lazy dogs.
//No spaces or newlines after that period
int main(int argc, char* argv[])
{
  String_Ptr hString = NULL;
  String_Ptr result = NULL;
  String_Ptr read = NULL;
  FILE* fp;

  /*************************************************/
  /***** Testing string_at and string_empty**********/
  hString = string_init_c_string("Hello World!");
  string_insertion(hString, stdout);

  printf("String size: %d \n", string_get_size(hString));
  printf("String inside hString: %s \n", string_c_str(hString));
  printf("The string has letter: %c at index: 10\n", *string_at(hString,6));
  printf("is the string empty:%d\n ", string_empty(hString));
  printf("\n");

  /*************************************************/
  /****Testing string_extraction ,string_insertion***/
```

```
  /*********Testing string_concat*******************/
  result = string_init_default();
  read = string_init_default();
  fp = fopen("simple.txt", "r");

  while(string_extraction(read,fp))
    {
      string_concat(result,read);
      string_insertion(result,stdout);
      printf("\n");
    }
  printf("\n");
/*************************************************/
  /*********Testing empty and pop_back*****************/
  while(string_empty(result) == FALSE)
    {
      string_pop_back(result);
    }
  printf("The string after popping all characters: ");
  string_insertion(result,stdout);
  printf("The string size: %d", string_get_size(result));
  printf("is the string empty: %d", string_empty(result));

  string_destroy(&result);
  string_destroy(&read);
  string_destroy(&hString);
  return 0;
}
```

Your output should be the following:

```
Hello World!
String size: 12
The string has letter: W at index: 6
is The string empty: 0

The
The quick
The quick brown
The quick brown fox
The quick brown fox jumped
The quick brown fox jumped over
The quick brown fox jumped over the
The quick brown fox jumped over the lazy
The quick brown fox jumped over the lazy dogs.

The string after popping all characters:
The string size: 0
is the string empty: 1
```

TA CHECKPOINT 1: Demonstrate to your TA that your functions behave as above. Show also that there is no memory leaks from your program by using valgrind.

What you will need to submit:

- string.c file holding the implementation of the above functions
- string.h file holder the functions declaration

- main_driver.c holding the above driver program
- A Makefile to use to compile your code
- A valgrind report showing that you do not have any memory leaks
- Combine all of the above files into a single compressed file. Name the compressed file: Lab6<first-initals><LastName>
- Submit the compressed file using the submit command to your lab TA.