Building a vector to hold strings

This lab assumes that you successfully finished all the previous labs from 1 to 6.

Lab details:

The goal of this lab is to build a vector holds strings. You have worked in the lecture on a vector that would hold integers and other types. Here the vector is used to hold and maintain strings. You will use the string object you developed in the other labs in this lab.

The definition for the vector_string struct is as follows:

```
struct vector_string
{
  int size;
  int capacity;
  String_Ptr * data;
};
typedef vector_string Vector_String, * Vector_String_Ptr;
```

This definition needs to be in the implementation file and not the header file. Your header file includes:

```
#ifndef VECTOR_STRING_H
#define VECTOR_STRING_H

#include "status.h"
#include "string.h"

typedef struct vector_string * Vector_String_Ptr;

// Precondition: None
// Postcondition: allocates memory to a pointer to a vector of strings
// it also sets the size to 0 and the capacity to 0
// it allocates an array of string_ptr of capacity 1
Vector_String_Ptr vector_string_init_default(void);


// Precondition: phVectorString: a pointer to a handle to a vector of strings
// Postcondition: makes sure that the memory is correctly freed
// You have multiple levels to free accordingly:
// 1- you need to loop through the array of String_Ptr and
// call the strong_destroy function
// 2- You need to free the array itself
// 3- You need to free the vector of strings (*phVectorString)
void vector_string_destroy(Vector_String_Ptr * phVectorString);

// Precondition: hVectorString: a handle to a vector of strings and
// hString: a handle to the string to be added. The string is already
// created and allocated by the calling program. Check if the hString
// is NULL before adding it to the vector.
// Postcondition: Adding a handle to a string to the end of the vector.
// If size == capacity of the vector (i.e, no room to add the string), then
// the vector needs to expand to hold more strings.
// Returns SUCCESS of the string was successfully added.
// Return  FAILURE:
```

```
// - hString was NULL or the hVectorString was NULL.
// - Could not expand the internal array
Status vector_string_push_back(Vector_String_Ptr hVectorString, String_Ptr hString);

// Precondition: hVectorString: a handle to a vector of strings to get its size
// Postcondition: the size of the vector of strings.
int vector_string_get_size(Vector_String_Ptr hVectorString);

// Preconiditon: hVectorString: a handle to a vector of strings to get its capacity
// Postcondition: the capacity of the vector of strings
int vector_string_get_capacity(Vector_String_Ptr hVectorString);

// Precondition: hVectorString: a handle to a vector of strings to check if
// empty or not
// Postcondition: True if it is empty, false otherwise
Boolean vector_string_empty(Vector_String_Ptr hVectorString);

// Precondition: hVectorString: a handle to a vector of strings and the index of the
// string to return
// Postcondition: a pointer to a string pointer. Returns null if the
// index is negative or > size.
String_Ptr* vector_string_at(Vector_String_Ptr hVectorString, int index);

#endif
```

You could use the vector object built in the lecture as a reference.

Afterwards, you need to add a driver program to test your functions with. Build one function at a time and test each one separately.

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
- vector_string.h file holding the above header file
- vector_string.c holding the implementation for the functions
- main_driver.c testing your functions
- A Makefile to use to compiler 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: Lab7<first-initals><LastName>
- Submit the compressed file using the submit command to your lab TA.