Dynamic Memory Allocation

On our Moodle homepage, you'll find a source file, readFile.c, along with an input file and an expected output file. You can also download these using the following curl commands:

curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise11/readFile.c
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise11/input.txt
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise11/expected.txt

The source file is a partial implementation of a program that reads a text file, storing its contents in memory. It will open an input file named "input.txt" and read all the characters from this file (up to the end-of-file). As it reads, it will store all the characters in a dynamically allocated string. After reading the whole file into the string, the program will print to standard output the total number of characters in the file, then the entire contents of the file. Most of this code has already been written for you. You just need to add the part to read characters and store them in one big string, and some code at the end to clean up. I've marked the places where you need to add code with ... in a comment.

You'll store all the characters in the array named buffer. This array starts out small, but you will include resizable array code to enlarge the buffer whenever it fills up. Implement the usual practice of doubling the buffer capacity each time you have to reallocate.

Your buffer needs to be a string (since we print it with "%s" at the end). So, you'll need to add a null terminator, right after the last character you read from the file. Be sure to plan for this null terminator in you array reallocation code. You need to have enough room to store all the characters from the input file, plus one more byte for the null terminator (the '\0') you'll add to mark the end.

You'll also need to add code to the end of the program to free the buffer and close the input file once you're done with them.

When you're done, your program will be able to read any text no matter how big it is. For example, you should be able to run it as follows to get it to read from the sample input file included with this exercise:

```
$ ./readFile
1286
This is a test input file for our resizable array exercise.
I want it to be long enough that we get a few reallocations while
<several lines deleted>
time on C++ at the end of the semester, but we'll just have time
to hit the highlights.)
```

When you're done, submit the source, readFile.c to the exercise_11 assignment on Moodle.