Lab 7 Instructions

Rob Hackman

Winter 2025

University of Alberta

Problem 1: Zip Strings

For this question you must write the function `zipStrings` which takes two string parameters of the same length and returns a new string that is the result of "zipping" them together.

If string `s1` is represented by the series of characters $c_1 c_2 ... c_n$ and string `s2` is represented by the series of characters $d_1 d_2 ... d_n$, then the result of zipping these two strings is the string $c_1 d_1 c_2 d_2 ... c_n d_n$.

Note: In order for your function to return a new, larger string, it must allocate memory using `malloc`

Zip Strings examples

- `zipStrings(‘‘abc’’, ‘‘xyz’’)` $\rightarrow$ `‘‘axbycz’’`
- `zipStrings(‘‘tiifn’’, ‘‘hssu!’’)`
  $\rightarrow$ `‘‘thisisfun!’’`

Problem 2: reading strings

You hopefully noticed in the starter code for `zipStrings` that the main function had code that looked like this:

```
char arr[256];
scanf("%s", arr);
```

And should have wondered - what happens if the string in the input stream is larger than 255 characters before the next whitespace? In this case a memory error would occur, as `scanf` would try and write past the end of our array.

Safer `scanf` string reading

As such, if using `scanf` to read a string the programmer should always specify the *maximum width* of the string to make sure that they don't incur a memory error.

```
char arr[256];
scanf("%255s", arr);
```

In this case, `scanf` will read until the next whitespace *or* until the maximum width is reached. However, this means if the actual string in the input stream *was* larger than your width you've only read *part* of that string!

You can see this issue by playing with `scanf_example.c` in the lab files.

Our own `readString`

Since `scanf` cannot be used to read an arbitrarily large string safely, we must write our function to do so. We will write the function `readString` which will have the following behaviour

- Reads in the next entire string from standard input, returning a pointer to a heap allocated string storing that data
- If the first non-whitespace character encountered is a double quote then your function will read all characters until the next double quote
- If the first non-whitespace character is any other character then your function will read until the next whitespace character or EOF
- If your function is unable to read a string it should return NULL

Hints

In order to read in a string of *any* length, your function will have to implement a growing array.

Make sure your program has no leaks! When you fail to read a string and return `NULL` make sure you don't leak any memory!