Lab #3

Summer 2024

Requirements

In this lab, you will cover pointer arithmetic and the casting of pointers. Be careful to note which type of pointer you are working with, and remember that different types may have different sizes. Also remember that you can treat characters like integers (IE: you can print them with %d to see their integer value).

1.1 strAlloc

```
char * strAlloc(int size)
```

Info: This function takes an integer representing the length of a string, and allocates a character array with the given size. The size of the array should be stored before the start of the array as an **int**. It returns a pointer to the array on success, or NULL on failure.

1.2 strLen

```
int strLen(char *str)
```

Info: This function takes a character array that was allocated using *strAlloc*, and returns the size which is stored before the array.

1.3 strCpy

```
void strCpy(char *dest, char *source)
```

Info: This function takes a source array, and a destination array, both of which were allocated using *strAlloc*. It will copy the contents of the source array into the destination array. It will require that the destination array is ***at least as large*** as the source array. For example:

```
// before
source = { 'H', 'E', 'L', 'L', 'O' };
dest =   { ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ' };
// after
dest =   { 'H', 'E', 'L', 'L', 'O', ' ', ' ', ' ', ' ' };
```

1.4 strRev

```
int strRev(char *dest, char *source)
```

Info: This function takes a source array, and a destination array, both of which were allocated using *strAlloc*. It will copy the contents of the source array into the destination array, in reverse order. It will require that the destination array is ***at least as large*** as the source array. It should only copy as much data as there is in the source array, and no more. it will return **1** if they are the same length, **0** if they are not, or **-1** in the case of some error. For example:

```
// before
source = { 'H', 'E', 'L', 'L', 'O' };
dest =   { ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ' };
// after
dest =   { 'O', 'L', 'L', 'E', 'H', ' ', ' ', ' ', ' ' };
// returns 0
```

(Note that the inverse of an index in an array is **size - index - 1**)

1.5 strFree

```
void strFree(char *str)
```

Info: This function takes a character array that was allocated using *strAlloc*, and frees the memory allocated to the array.

Submission Information

Submit this assignment by using the mucsmake command.

Use the following command on Hellbender:

mucsmake <course> <assignment> <filename>

For example:

mucsmake 2050 lab3 lab3.c

Rubric: 18 points

1. Write required *strAlloc* function
   * 4 points
2. Write required *strLen* function
   * 2 points
3. Write required *strCpy* function
   * 3 points
4. Write required *strRev* function
   * 6 points
5. Write required *strFree* function
   * 3 points

Notice:

1. All of your lab submissions **must** include documentation in the form of code comments to receive full points. In addition, your program is expected to have a **comment header** at the top that includes your name, pawprint, the course you are taking, and the lab that you solved. You can refer to the Lab 0 document for an example of the comment header.
2. All of your lab submissions must compile under GCC using the -Wall and -Werror flags to be considered for a grade. These flags will automatically be applied if you use the compile command.
3. Do **NOT** change the given function prototype or anything else in the provided .h file.