CptS 122 – Data Structures
January 12, 2024

Your Name: _____ TA's Name: _____
ID#: _____ Section #: _____

SOLUTION: Take Home: Quiz 1 (15 pts) – Review of C Language Concepts

Using Canvas https://canvas.wsu.edu/, please submit your solution to the correct quiz folder. Your solution should be a .pdf file with the name `<your last name>_quiz1.pdf` and uploaded. To upload your solution, please navigate to your correct Canvas ***lab*** course space. Select the "Assignments" link in the main left menu bar. Navigate to the correct quiz submission folder. Click the "Start Assignment" button. Click the "Upload File" button. Choose the appropriate .pdf file with your solution. Finally, click the "Submit Assignment" button.

1. (5 pts) Write/Implement a function definition for a function `my_strncpy()` with the following header:

```
char *my_strncpy (char *destination, const char *source, int n)
```

This function copies no more than `n` characters from the string pointed to by `source` to the buffer pointed to by `destination`. If the length of the C string in `source` is less than `n`, then the `destination` is padded with null characters until `n` characters have been copied to it. The function returns `destination`. **NOTE: you must use array notation in your definition.**

```
/* This function copies no more than n characters from the string pointed to by
source to the buffer pointed to by destination. If the length of the C string in
source is less than n, then the destination is padded with null characters until n
characters have been copied to it. The function returns destination. This
implementation uses array notation. */

char *my_strncpy(char *destination, const char *source, int n)
{
	int index = 0;

	for (index = 0; (source[index] != '\0') && (index < n); ++index)
	{
		// copy one character at a time
		destination[index] = source[index];
	}

	// if fewer than n characters were copied, then
	// assign the null character to pad end of destination
	for (; index < n; ++index)
	{
		destination[index] = '\0';
	}

	return destination;
}
```

Your Name: _____ TA's Name: _____
ID#: _____ Section #: _____

2. (5 pts) Write/Implement a function definition for a function `my_strncat()` with the following header:

```
char *my_strncat (char *destination, const char *source, int n)
```

This function appends no more than `n` characters from the string pointed to by `source` to the end of the string pointed to by `destination`. The null character is appended to the end of the result. If the length of the C string in `source` is less than `n`, then only the content up to the terminating null character is copied. The `destination` pointer is returned. **NOTE: you must use pointer arithmetic and notation in your definition.**

```
/*This function appends no more than n characters from the
string pointed to by source to the end of the string pointed
to by destination. The null character is appended to the end of
the result. If the length of the C string in source is less than n,
then only the content up to the terminating null character is copied.
The destination pointer is returned. This implementation uses
pointer arithmetic.*/

char *my_strncat(char *destination, const char *source, int n)
{
	int destIndex = 0, srcIndex = 0;

	// find the end of the destination string, so that we
	// can overwrite the destination string's null character
	// and append the source string starting form there.
	for (destIndex = 0; (*(destination + destIndex) != '\0'); ++destIndex);

	// note: we don't want to reset destIndex back to 0! Start from
	// end of destination!
	for (srcIndex = 0; (*(source + srcIndex) != '\0') && (srcIndex < n);
	     ++srcIndex, ++destIndex)
	{
		// copy one character at a time
		*(destination + destIndex) = *(source + srcIndex);
	}
	// assign the null character
	*(destination + destIndex) = '\0';

	return destination;
}
```

Your Name: _____ TA's Name: _____
ID#: _____ Section #: _____

3. (5 pts) Write a function `strlen_recursive ()` which accepts a *pointer* to a string and recursively *counts* the number of *characters* in the provided string, *excluding* the null character. This function should return the *number* of characters in the string.

```
int strlen_recursive(char *str)
{
    int len = 0;

    // recursive step?
    if (*str != '\0') // yes, it's time to do the recursive step
    {
           // count the character, and go to the next memory
              // location in the string
           len = 1 + strlen_recursive (str + 1);
    }

    return len; // when the null character is encountered len is 0, so doesn't
                  // impact cummulative count for len
}
```