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

Using Canvas, submit your quiz. You will submit your assignment in the ***lab*** Canvas space. Upload your solution to the appropriate “Quiz” space. You must upload your solution named <your last name>\_quiz1.pdf by the due date and time. The first quiz is due before your Lab 2 starts next week.

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.**

Solution:

char\* my\_strncpy(char\* destination, const char\* source, int n)

{

int i = 0;

while (i < n)

{

if (source[i] != '\0')

{

destination[i] = source[i];

}

else

{

destination[i] = '\0';

}

i++;

}

//In case the source string has n or more than n characters, we need to make the ending NULL character for the destination string

if (destination[n - 1] != '\0')

{

destination[n] = '\0';

}

return destination;

}

1. (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.**

Solution:

char\* my\_strncat(char\* destination, const char\* source, int n)

{

int i = 0;

//First, we find the position of the NULL character at the end of the string pointed to by destination

for (i = 0; \*(destination + i) != '\0'; i++)

{

}

int last\_post = i; // i is currently the position of the NULL character at the end of the string pointed to by destination

//We start append a copy of the string pointed to by source to this position

for (i = 0; \*(source + i) != '\0'; i++)

{

\*(destination + last\_post + i) = \*(source + i);

}

i = 0; //Reset i

while (i < n && source[i] != '\0')

{

\*(destination + last\_post + i) = \*(source + i);

i++;

}

\*(destination + last\_post + i) = '\0';

return destination;

}

1. (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.

Solution:

int strlen\_recursive(const char\* str)

{

if (\*str == '\0')

{

return 0;

}

else

{

return (1 + strlen\_recursive(++str));

}

}