THE UNIVERSITY OF SYDNEY

INFO1910 Week 8 Tutorial

In this tutorial you will be writing a small c library for various functions related to c strings. Recall that a c string is an array of characters, ending in a null terminator character, which in c source code is written \0. The null terminator is not considered part of the string's length.

Question 1

Write a function which finds the length of a given string.

```
int string_len(char* str)
```

Question 2

Write a contains_substring function. It receives a primary string and a secondary string, and finds whether or not the secondary string is present within the primary one. If so, the function returns the first index at which the secondary string appears within the primary string. If not, the function returns -1.

```
int contains_substring(char* primary, char* secondary)
```

Question 4

Write a string_equal function. It receives two strings, returns 1 if they are identical, and 0 otherwise. There are at least two ways you could write this function; in addition to the direct comparison via loops, think about how you could use the contains_substring function from the previous question to do the task. Which do you think is a more efficient method?

```
int string_equal(char* str1, char* str2)
```

Question 3

Write a string_copy function. It receives a string, and a character array into which the string will be copied. The function also receives an integer giving the length of the array which will receive the copy; do not copy beyond the end of the array given. Don't forget to write the null terminator at the end of the copied string!

```
void string_copy(char* str, char* buffer, int buffer_len)
```

Question 5

Write a strip function to mimic the strip function in python. It receives a string to read from, a buffer to create a new string in, and a character which should be removed from the left and right sides of the given string. Note that if the character to strip appears multiple times in a row on one side before other characters appear, all instances of it should be removed. The function may assume that the given buffer is at least long enough to contain the original string (make sure you provide a buffer of sufficient size in testing; there needs to be a space for the null terminator too). The newly created string should begin at the start of the given buffer, not part of the way in.

```
void strip(char* str, char* buffer, char to_strip)
```

Question 6

Write a split function to mimic the split function in python. It receives a string, a buffer to write new strings into, an array of character pointers, and a delimiter character. The function should create several strings (using the given buffer as working space) based on the original string. A new string begins each time the delimiter character is encountered. If the delimiter is encountered twice in succession, there will be an empty string in the resultant array. Pointers to the beginnings of each of the newly created strings should be written into the given string array (array of character pointers), so that it may be treated as an array of the strings resulting from the split operation by the calling function. The function should return the total number of strings in the resultant string array. Hint: think about how long the buffer will need to be to guarantee there is enough space for this function to work.

```
int split(char* str, char* buffer, char** array, char delim)
```

Question 7

Write a join function to mimic the join function in python. It receives an array of strings, a buffer of characters to write into, an integer giving the number of strings in the array, and a delimiter character. The function will concatenate all the strings in the string array (in order), placing a single delimiter

character at each join point. This single new string will be written into the given buffer (don't forget to include the null terminator). Again, think about how long the buffer will need to be to guarantee this works.

```
void join(char** strings, char* buffer, int num_strings, char delim)
```

Question 8

If you have been using a main function in the same source code file to test your functions, remove it. Create a header (.h) file declaring the existence of the functions in this file. Practise writing a separate main function in a separate file, including the header, and compiling the string library along with your main program file so that the functions you have written can be used in other programs you may write in the future.