COMP1511 18s2

Week-06 Tutorial Exercises

1. .

```
#include <stdio.h>

int main(void) {
    char str[10];
    str[0] = 'H';
    str[1] = 'i';
    printf("%s", str);
    return 0;
}
```

   a. What will happen when the above program is compiled and executed?
   b. How do you correct the program.

2. Let's assume that `fgets` statement reads "Hi Sydney" from stdin and stores it in `line` (character array), see below.

```
line = ['H', 'i', ' ', 'S', 'y', 'd', 'n', 'e', 'y' , '\n', '\0' , 'p', '1', '-', 'c']
```

   - How can you calculate a length of a string?
   - What is the length of the string starting at index 0?
   - What is the last character in the string starting at index 0? Why?
   - How can we make sure that the last character of the string is 'y' (last char of 'Sydney')?
   - After the modification, what is the length of the string?
   - How can you print the string in a reverse order, for example "yendyS iH" .

3. Name 3 errors in this program:

```
#include <stdio.h>

#define MAX_LINE 4096

int main(void) {
    char line[MAX_LINE];
    int  i;

    while (fgets(line, MAX_LINE, stdin) != NULL) {
        i = MAX_LINE;
        while (line[i] != '\n') {
            i = i - 1;
        }
        printf("the line is %d characters long\n", i);
    }
    return 0;
}
```

4. Write a program `line_length.c` which reads lines from its input and prints how many characters each line contains.

   The only functions you can use are `fgets` and `printf`.

   You can assume lines contain at most 4096 characters.

   For example:

```
$ ./line_length
Andrew Rocks
line 12 characters long
A very long line.
line 17 characters long
short
line 5 characters long

line 0 characters long
```

5. What do you use **fopen** for and with what parameters?

6. Under what circumstances would **fopen** return NULL?

7. Write a C code to read in the first line of a file named **line_leng** and display it on the screen.

8. Write C code to read in a string from a user and write it to a file called **data.txt**. If the file **data.txt** exists, it should be overwritten

9. Write C to read in a string from a user and write it to a file called **data.txt**. If the file **data.txt** exists, it should append the line of text to the end of the file.

10. Write a C program `reverse.c` which reads lines and writes them out with the characters of each line in reverse order. It should stop when it reaches the end of input.
    For example:

```
$ ./reverse
The quick brown fox jumped over the lazy dog.
.god yzal eht revo depmuj xof nworb kciuq ehT
It was the best of times. It was the worst of times.
.semit fo tsrow eht saw tI .semit fo tseb eht saw tI
This is the last line.
.enil tsal eht si sihT
<control-d>
```

11. Write a C function that is given two strings and returns 1 if the first begins with the second (and 0 otherwise). For example, "APPLE" begins with "APP".

```
int beginsWith(char *string1, char *string2);
```

Revision questions

The remaining tutorial questions are primarily intended for revision - either this week or later in session.
Your tutor may still choose to cover some of the questions time permitting.

12. Write a program `file_max.c` that reads a set of integer numbers from a file and prints out the maximum number to standard output. The name of the input file should be specified as a command line argument.

13. Write a program `strip_comments.c` which reads lines from its input and prints them after removing any C // style comments.
    In another words if the line contains // it does not print the // or anything after it.
    The only functions you can use are `fgets` and `printf`.

    You can assume lines contain at most 4096 characters.

    For example:

```
$ ./strip_comments
    x = x + 1;  // This means add one to the variable x
    x = x + 1;
```

    Also - is that a good comment to add to a C program?

14. Write a program `filter_empty_lines.c` which reads lines from its input and prints them only if they contain a non-white-space-character.
    In another words remove lines are empty or contain only white-space.

    The only functions you can use are `fgets` and `printf`.

    You can assume lines contain at most 4096 characters.

    You can assume there are only 3 white space characters, space, tab & new-line.

    For example:

```
$ ./filter_empty_lines
full line
full line

another no-empty line
another no-empty line
```

15. Strlen is a function that returns the length of a string. Write your own C function to do the same.

```
int myStrlen(char *string);
```

16. Write a C function that is given two strings and returns 1 if they are both equal and 0 otherwise. This is a simplified version of strcmp, provided in the C library. Try writing one yourself.

```
int myStrCmp(char *string1, char *string2);
```

17. Write a C function that is given two strings and returns 1 if the first is a substring of the second (and 0 otherwise). For example, "APP", "E", "PL" are all substrings of "APPLE".

```
int isSubstring(char *substring, char *string);
```