COMP9024 25T3
Dr. Aditya Joshi

Week 2 Practical
Elementary Data and Control Structures in C

1. (String length)

   You are to implement the `mystrlen` function described in Week 2 Tutorial, Exercise 4 and declared in `mystrlen.h`.

   `mystrlen.h`

   ```
   #define MAXLEN 128

   /**
    * @brief Calculates the length of a string.
    *
    * The mystrlen() function calculates the length of the string pointed to by s,
    * excluding the terminating null byte ('\0').  If no null byte is found within
    * the first MAXLEN bytes of the string, the function returns MAXLEN.
    *
    * @param s Pointer to the first character of a string.
    * @return  The length of the string s, or MAXLEN if no null byte found within
    *          MAXLEN characters.
    */
   int mystrlen(char *s);
   ```

   - The function **must** be implemented in a file `mystrlen.c`.
   - `mystrlen.c` **must** `#include "mystrlen.h"`.
   - `mystrlen.c` **must not** `#include` any files other than `mystrlen.h`.
   - `mystrlen.c` **must not** define any functions other than `mystrlen`.
   - `mystrlen.h` **must not** be modified.

   We have created a script that can automatically test your function. To run this test you can execute the `dryrun` *program that corresponds to this exercise. It expects to find a file named* `mystrlen.c` *in the current directory. You can use dryrun as follows:*

   ```
   prompt$ 9024 dryrun mystrlen
   ```

2. (String length client)

   You are to implement a program `lencalc.c` that uses the `mystrlen` function to calculate the length of a string.

   The program must repeatedly prompt the user to input a string and then print the length of the string, as determined by the `mystrlen` function.

   Should the user simply press the `RETURN` key, without entering any additional characters, then the program should print "Goodbye!" and terminate.

   The program **must** `#include` two files and two files only:

   1. `<stdio.h>`: for the `printf` and `fgets` functions
   2. `"mystrlen.h"`: for the `mystrlen` function

   Starter code for `lencalc.c` is provided below. Notice we use `fgets()` to read the user input. `scanf()` can work well for formatted input, but as we're reading a random string of arbitrary length, `fgets()` is a safer choice. In the `fgets()` call below, we're reading up to `BUFSIZ` characters from `stdin` (i.e. the keyboard) and storing it in the character array `s`.

lencalc.c

```
#include <stdio.h>
#include "mystrlen.h"

int main(void) {
    char s[BUFSIZ];
    int len;

    // Modify this section of the code
    // -------------------------------

    printf("Enter a string: ");
    fgets(s, BUFSIZ, stdin);
    printf("You entered: %s", s);

    // -------------------------------

    return 0;
}
```

An example of your program executing is below, with user input highlighted in bold:

```
prompt$ ./lencalc
Enter a string: hello
6
Enter a string: hello world
12
Enter a string:
Goodbye!
prompt$
```

We have created a script that can automatically test your program. To run this test you can execute the `dryrun` *program that corresponds to this exercise. It expects to find a file named* `lencalc.c` *in the current directory. You can use dryrun as follows:*

```
prompt$ 9024 dryrun lencalc
```

Due Date

Monday, 29 September, 16:59:59. Late submissions will not be accepted.

Submission

⚠️ **Important: Every new submission completely replaces your previous one. You must upload all files to be marked each time you submit.**

You should submit your files using the following `give` command:

```
prompt$ give cs9024 week2 mystrlen.c lencalc.c
```

Alternatively, you can select the option to "Make Submission" at the top of this page to submit directly through WebCMS3.

Important notes:

- Make sure you spell all filenames correctly.
- You can run `give` multiple times. Only your last submission will be marked.
- Every new submission completely replaces your previous one. You must upload all files to be marked each time you submit.
- Whether you submit through the command line or WebCMS3, it is your responsibility to ensure it reports a successful submission. Failure to submit correctly will not be considered as an excuse.
- You cannot obtain marks by e-mailing your code to tutors or lecturers.

Assessment

- Each exercise is worth 1 mark, for a total of 2 marks.
- Submissions will be auto-marked shortly after the deadline.
- It is important that the output of your program follows exactly the format of the sample output, otherwise auto-marking will result in 0 marks.
- Ensure that your program compiles on a CSE machine with the standard options, i.e. `-Wall -Werror -std=c11`. Programs that do not compile will receive 0 marks.
- Auto-marking will use test cases different to `dryrun`. (Hint: do your own testing in addition to running `dryrun`).

Collection

Once marking is complete you can collect your marked submission using the following command:

```
prompt$ 9024 classrun -collect week2
```

You can also view your marks using the following command:

```
prompt$ 9024 classrun -sturec
```

You can also collect your marked submission directly through WebCMS3 from the "Collect Submission" tab at the top of this page.

Plagiarism

Group submissions will not be allowed. Your programs must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (including submissions for similar assessments in previous years, if applicable) and serious penalties will be applied, including an entry on UNSW's plagiarism register.

You are also not allowed to submit code obtained with the help of ChatGPT, GitHub Copilot, Gemini or similar automatic tools.

- ***Do not copy ideas or code from others***
- ***Do not use a publicly accessible repository or allow anyone to see your code***
- ***Code generated by ChatGPT, GitHub Copilot, Gemini and similar tools will be treated as plagiarism.***

Please refer to the on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Academic Integrity and Plagiarism
- UNSW Plagiarism Policy