CS101 Lab-8

Fundamentals of Computer Programming

Shreya Agarwal & Rachit Nimavat

In this lab, we will work with `clock` and `time` functions from `time.h`

Background:

- **`time(NULL)`**: Returns the current calendar time as a `time_t` value (usually seconds since Jan 1, 1970). Its most common use in small programs is to provide a different "seed" for the random number generator, ensuring you get a different random sequence every time you run the program: `srand(time(NULL));` Note that `srand` is included in `stdlib.h`
- **`clock()`**: Returns the number of clock "ticks" that have passed since the program started. This is a measure of processor time used, not calendar time. Notice that its return type is `clock_t` which can be safely typecasted to long int in our labs.

Instructions for the lab assignment:

Solve the **first three problems** in the lab. The other problems are for homework.

1. Write a function that swaps two integers when passed by reference. Is it possible to write a function that swaps them when passed by value?

2. Write a recursive function to find the sum of all digits in a `long int` number. Note that the input number can possibly be negative.

3. Create a simple typing test. Define a list of test strings (that do not contain new lines themselves). Use `time(NULL)` to seed the random number generator and save `time_t start = time(NULL)`. Pick a random string from the list. Display the chosen string and ask the user to type it *exactly* as shown. After the user types the string, save `time_t end = time(NULL)` and use `difftime(end, start)` to measure how long the user takes to type the string and press Enter. Compare the user's input with the original string. Print whether the user was correct or incorrect, and how many seconds they took. Calculate WordsPerMinute. Modularize your code with the help of appropriate functions.

4. Create a function `void censorString(char str[], const char *badWord).` It should find all case-insensitive occurrences of the `badWord` within `str` and replace each character of that word with an asterisk (`*`). Notice that you should censor the `badWord` even if it appears as a proper substring of a larger word.

5. Write a function `void scaleArray(int arr[], int size, int factor)`. This function should loop through the array and multiply every element by the factor. Since arrays are passed by reference, this will modify the original array in main. Is specifying `size` necessary as a function argument? Can't you get the size by writing `sizeof(arr)/sizeof(arr[0])`?

Solution for Lab-7 Problem-1

```c
#include <stdio.h>
#include <string.h>

#define MAX_TEXT_LEN 1000
#define BUFFER_SIZE 1002

int main() {
    char input[BUFFER_SIZE];
       printf("Enter a line of text (max %d characters): ",
MAX_TEXT_LEN);

    // fgets() returns NULL on error
    if (fgets(input, BUFFER_SIZE, stdin) == NULL) {
        printf("Error reading input.\n");
        return 1; // Exit with on error
    }

     /* Check if the input was too long. If fgets() filled the
buffer *before* finding a newline it means the line was longer
than the buffer could hold. Thus, we check for the presence of
'\n' in input.
    */
    int i=0;
    while(input[i] != '\0' && input[i] != '\n') i++;
    if(input[i] == '\0') {
        //didn't find newline
        printf("Input string is too long...\n");
        return 1;
    } else {
        //input[i] is \n
           input[i] = '\0'; //replace '\n' with'\0' to properly
terminate the string
        //i now becomes the length of input string
        printf("Manually calculated string length is %d\n", i);
        printf("Length (using strlen): %d\n", strlen(input));
    }
    return 0;
}
```