CS101 Lab-7

Fundamentals of Computer Programming

Shreya Agarwal & Rachit Nimavat

We are halfway through the CS101 course. As part of the labs so far, we have gained experience in Linux command-line operations, covering file management (e.g., creating `.c` files)(`gedit`), compilation(`gcc`), and execution(`./a.out`). We explored the application of regular expressions, the grep utility, and piping (`|`) for data manipulation. We have studied how to redirect input from a command (`|`), a file (`<`), how to redirect output to a file (`>`) and, `chmod` command for the modification of file permissions. We now shift focus to string manipulation within the C programming language. Recall that a string is defined as a null-terminated array of characters, where the null character (`\0`) signifies the string boundary.

Instructions for the lab assignment:

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

1. Write a program to read a full line of text (including spaces) from the user using **fgets()**. If the input text length is more than 1000, display an error message. If not, print the string and its total length. (Calculate and print the string's length using both the **strlen()** function and a custom loop that counts characters until the null terminator `\0` is encountered.)
2. Write a program to take a long string (haystack) and a small string (needle). Use **strstr()** to locate the first occurrence of the needle in the haystack. Report the starting index.

3. Write a program to reverse a string without using a second temporary array. Use two pointers (one at the start, one at the end) and a single temporary variable for swapping characters.
4. Write a program that checks if a user-input string is a palindrome. Ignore spaces, punctuation, and case during the check.
5. Find the most frequently occurring character in a string. Use a frequency array (e.g., an array of size 256) to store character counts.

Solution for Lab-6 Problem-1

```c
#include <stdio.h>

int main() {
    int sz;
    printf("Enter array size: ");
    scanf("%d", &sz);
    if(sz < 0) {
        printf("Invalid size.\n");
        return 1;
    }

    int arr[sz], target; //VLA declaration. Will not work if sz is too large.
    printf("Enter %d integers: ", sz);
    for(int i=0; i<sz; i++) scanf("%d", &arr[i]);
    printf("Enter target: ");
    scanf("%d", &target);

    for(int i=0; i<sz; i++) {
        for(int j=i+1; j<sz; j++) {
            if(arr[i]+arr[j]==target) {
                //Will not work if arr[i] and arr[j] are so large that arr[i]+arr[j] overflows
                printf("Numbers at indices %d and %d: %d+%d=%d\n", i, j, arr[i], arr[j], target);
                return 0;
            }
        }
    }

    printf("No numbers of array sum up to %d\n", target);
    return 0;
}
```

Solution for Lab-6 Problem-5

```c
#include <stdio.h>
#define ARRAY_SIZE(arr) (sizeof(arr) / sizeof((arr)[0]))

int main() {
      int arr[] = {2, 3, 4, 3, 7, 9, 0};
      int target = 8;
      int sz = ARRAY_SIZE(arr);   //assume sz<=20.
      // Loop from 0 to 2^sz - 1. Each 'mask' value represents one subset.
      for(int mask=0; mask<(1<<sz); mask++) {
          //if ith bit of mask is 1, we include that number, otherwise not
          long int sum=0;
          for(int i=0; i<sz; i++) {
              if(mask&(1<<i)) sum += arr[i];
          }
          if(sum==target) {
              printf("The following elements sum up to %d\n", target);
              for(int i=0; i<sz; i++) {
                  if(mask&(1<<i)) printf("%d ", arr[i]);
              }
              return 0;
          }
      }
      printf("No elements sum up to %d\n", target);
      return 0;
}
```