LAB REPORT

CSE332: Compiler Design Lab

02

Topic: Solving String Problem Using C.

Submitted To

Shadman Rabby (SHR)

Lecturer

Department of CSE,

Daffodil International University

Submitted By

Student ID: 221-15-5400

Section: 61_A2

Student Name: M. B. Mahir Tanzim

Date of Assignment Distribution : 31 January 2025

Date of Assignment Submission: 5 April 2025

Experiment No: 02		**Mapping:** CO1 and CO2
Experiment Name	Solving String Problem Using C	

Experiment Details:

Problem 01: Write a program that will **count vowel, consonant, and digit** from a given string **.**

Solution:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main(){
    char s[200];
    gets(s);
    int cnt_d = 0, cnt_v = 0, cnt_c = 0;
    for(int i = 0; s[i] != '\0'; i++){
        s[i] = tolower(s[i]);
        if(s[i] >= '0' && s[i] <= '9')
            cnt_d++;
        else if(s[i] >= 'a' && s[i] <= 'z'){
            if (s[i] == 'a' || s[i] == 'e' || s[i] == 'i' ||
s[i] == 'o' || s[i] == 'u')
                cnt_v++;
            else
                cnt_c++;
        }
    }
    printf("Digits: %d\n", cnt_d);
    printf("Vowels: %d\n", cnt_v);
    printf("Consonant: %d\n", cnt_c );

}
```

Problem 02: Write **two** C program that will **tokenize a string. (using strtok() and also without using any library function)**

Solution (using strlock()):

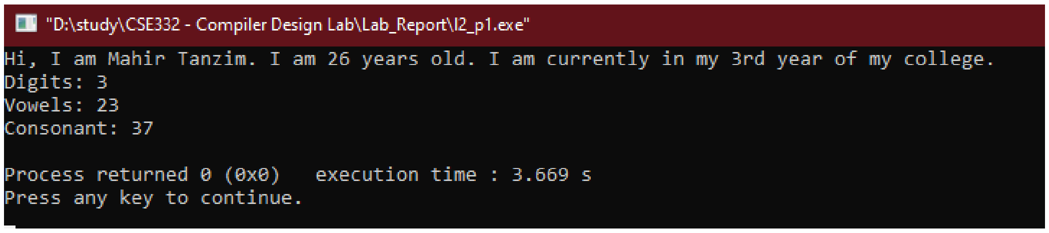
```
#include <stdio.h>
#include <string.h>
int main() {
    char s[100], *token;
    printf("Enter a string: ");
    gets(s);

    token = strtok(s, " \n");
    while (token != NULL) {
        printf("%s\n", token);
        token = strtok(NULL, " \n");
    }

    return 0;
}
```

Solution (without strlock()) :

```
#include <stdio.h>
int main() {
    char s[100];
    printf("Enter a string: ");
    gets(s);
    int i = 0, start = 0;
    while (s[i] != '\0') {
        if (s[i] == ' ' || s[i] == '\n') {
            s[i] = '\0';
            printf("%s\n", &s[start]);
            start = i + 1;
        }
        i++;
    }
    if (start < i) {
        printf("%s\n", &s[start]);
    }
    return 0;
}
```

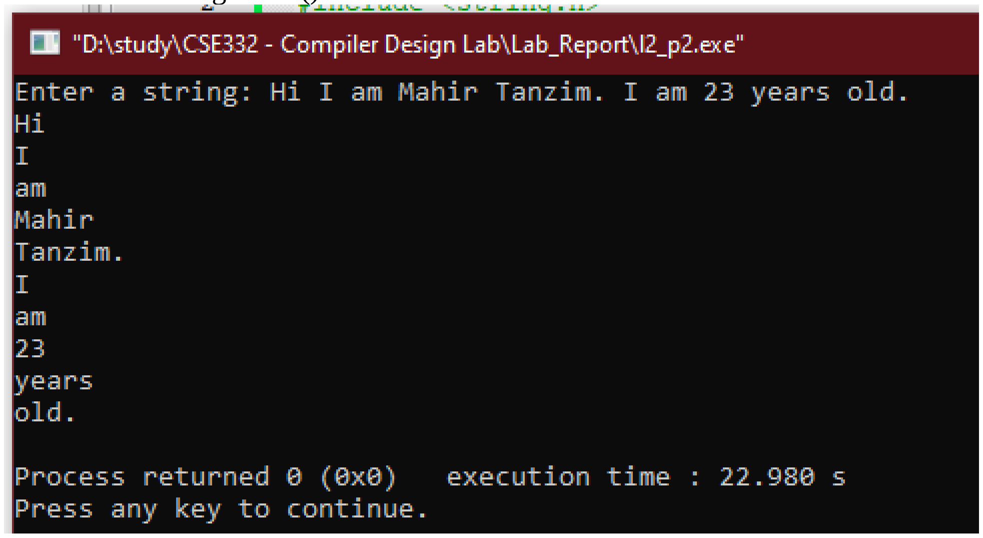
Obtained Output:

Problem 01:



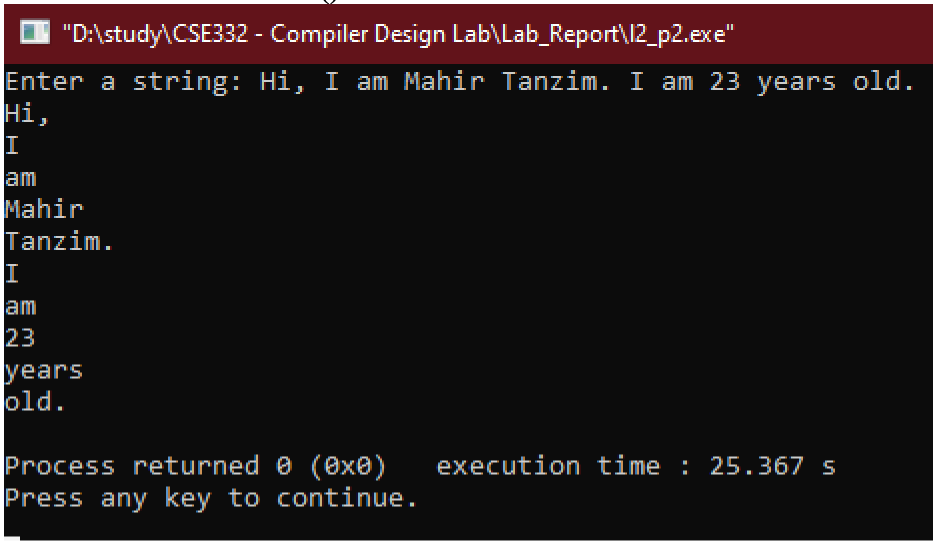
```
"D:\study\CSE332 - Compiler Design Lab\Lab_Report\l2_p1.exe"
Hi, I am Mahir Tanzim. I am 26 years old. I am currently in my 3rd year of my college.
Digits: 3
Vowels: 23
Consonant: 37

Process returned 0 (0x0)   execution time : 3.669 s
Press any key to continue.
```

Problem 02: Using strtok()



```
"D:\study\CSE332 - Compiler Design Lab\Lab_Report\l2_p2.exe"
Enter a string: Hi I am Mahir Tanzim. I am 23 years old.
Hi
I
am
Mahir
Tanzim.
I
am
23
years
old.

Process returned 0 (0x0)   execution time : 22.980 s
Press any key to continue.
```

Problem 03: without strtok()



```
"D:\study\CSE332 - Compiler Design Lab\Lab_Report\l2_p2.exe"
Enter a string: Hi, I am Mahir Tanzim. I am 23 years old.
Hi,
I
am
Mahir
Tanzim.
I
am
23
years
old.

Process returned 0 (0x0)   execution time : 25.367 s
Press any key to continue.
```

Desired Output?
YES

Alternative Steps/Solution (If any):

- Instead of using **`strtok(),`** we can manually traverse the string and extract words using a separate buffer.
- Using **`sscanf()`** for tokenization can be an alternative method.
- Implementing a dynamic approach with **`malloc()`** can handle variable-length strings efficiently.

Observation/ Comments:

Both methods successfully tokenize the string. **`strtok()`** provides an easier implementation, but it modifies the original string. The manual approach gives more control but requires careful string manipulation. Using **`sscanf()`** or a buffer-based approach can offer additional flexibility.