Lab Report

Course title: Compiler design lab

Course Code: CSE 332

Submitted To:

Shah Md. Tanvir Siddique

Assistant Professor

Department of Computer Science and Engineering

Daffodil International University

Submitted BY:

Tasmin Karim

ID: 193-15-2920

Section: PC-B

Submission Date:

28 May 2023

Index

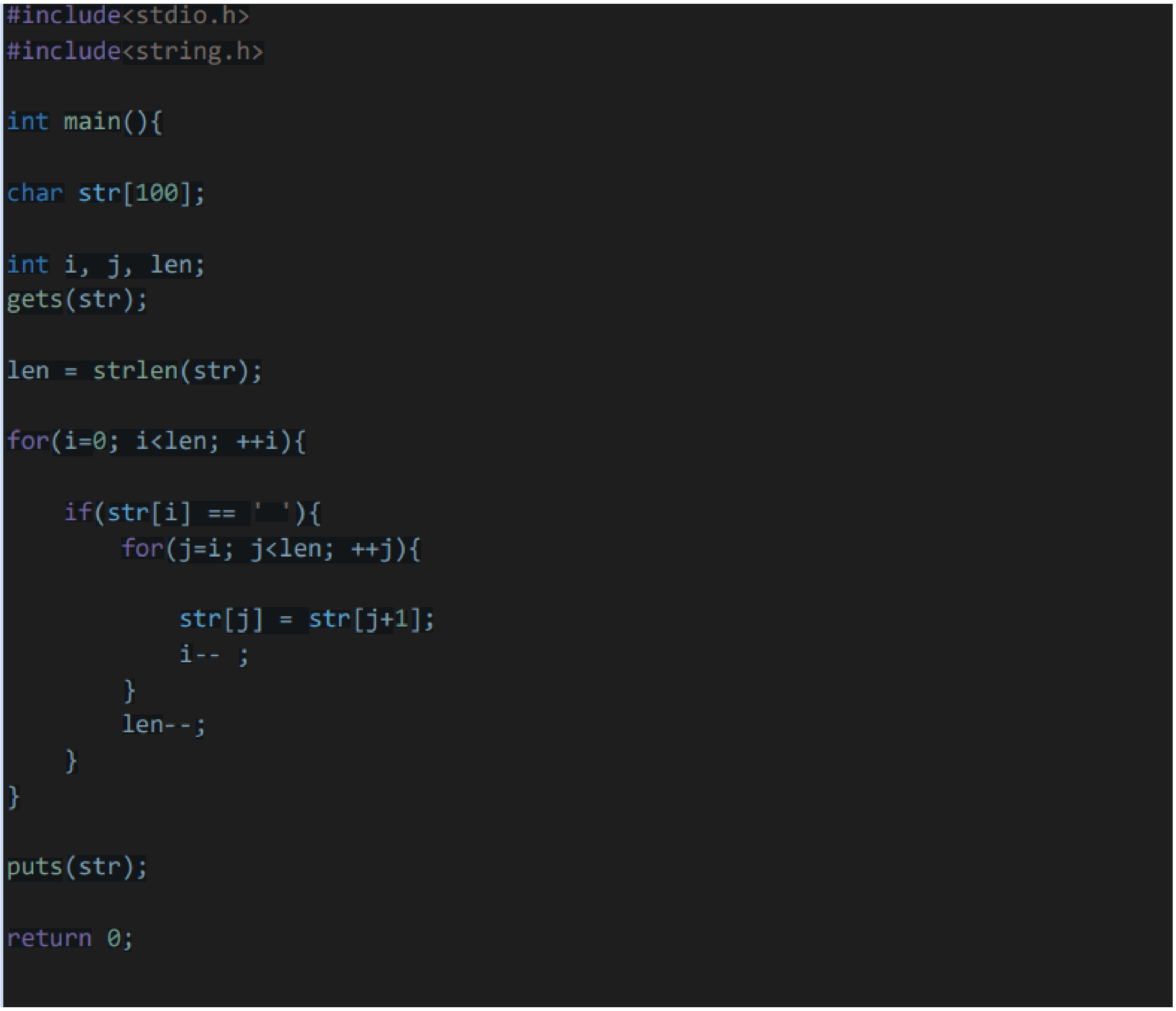
Problem 01:

Description: Write a Program to recognize multiple white spaces and remove it.

Sample Input: int a = 5

Sample Output: inta=5

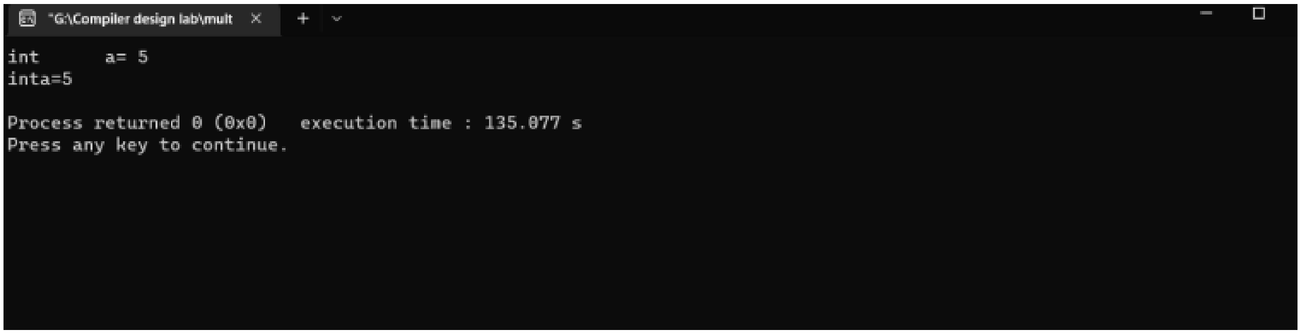
Code:



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

int main(){

char str[100];

int i, j, len;
gets(str);

len = strlen(str);

for(i=0; i<len; ++i){

    if(str[i] == ' '){
        for(j=i; j<len; ++j){

            str[j] = str[j+1];
            i-- ;
        }
        len--;
    }
}

puts(str);

return 0;
```

```
}
```

Output:



```
int      a= 5
inta=5

Process returned 0 (0x0)   execution time : 135.077 s
Press any key to continue.
```

Problem 02:

Description: Write a Program to recognize whether its a Single Line Comment or Multiline Comment.

Sample Input: //I am a programmer

Sample Output: Single Line Comment!

Sample Input: /*Multiline example*/

Sample Output: Multiline Comment!

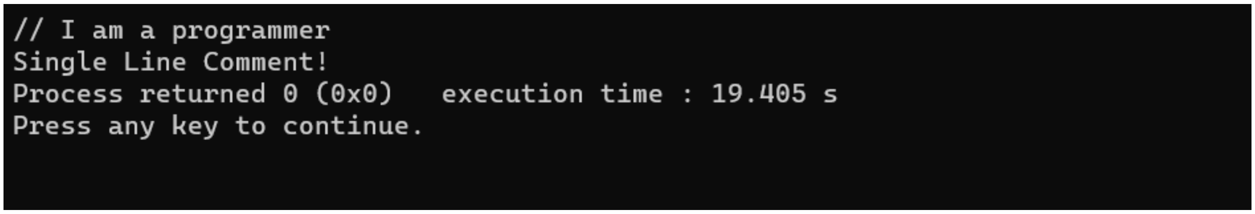
Sample Input: /*Not a comment

Sample Output: Not a comment!

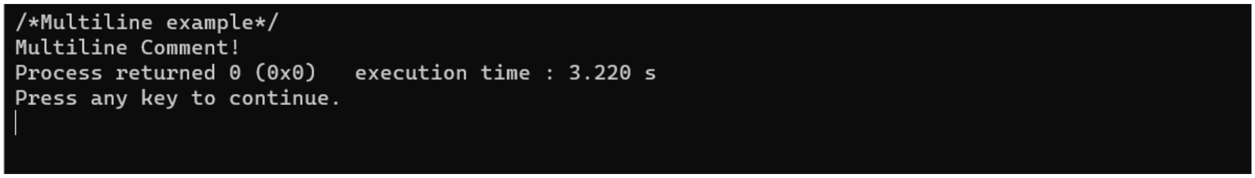
Code:

```
#include<stdio.h>

#include<string.h>

int main(){

char str[100];

int i;

gets(str);

int len = strlen(str);


if(str[0]=='/' && str[1]=='/')

    printf( "Single Line Comment!");

else if (str[0]=='/'&& str[1]=='*' && str[len-2]=='*'&& str[len-1]=='/')

    printf("Multiline Comment!");

else

    printf("Not a comment!");

return 0;

}
```

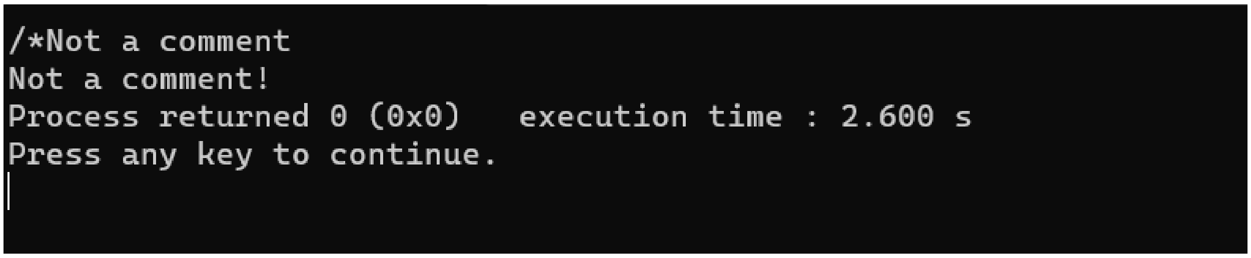
<u>Output:</u>



```
// I am a programmer
Single Line Comment!
Process returned 0 (0x0)   execution time : 19.405 s
Press any key to continue.
```



```
/*Multiline example*/
Multiline Comment!
Process returned 0 (0x0)   execution time : 3.220 s
Press any key to continue.
```



```
/*Not a comment
Not a comment!
Process returned 0 (0x0)   execution time : 2.600 s
Press any key to continue.
```

Problem 03:

Description: Write a Program to recognize string which is in inside the input.

Input: Hello world "life is beautiful" welcome.

Output will show the string only.

Output: Input contains string "life is beautiful"

What's a string?

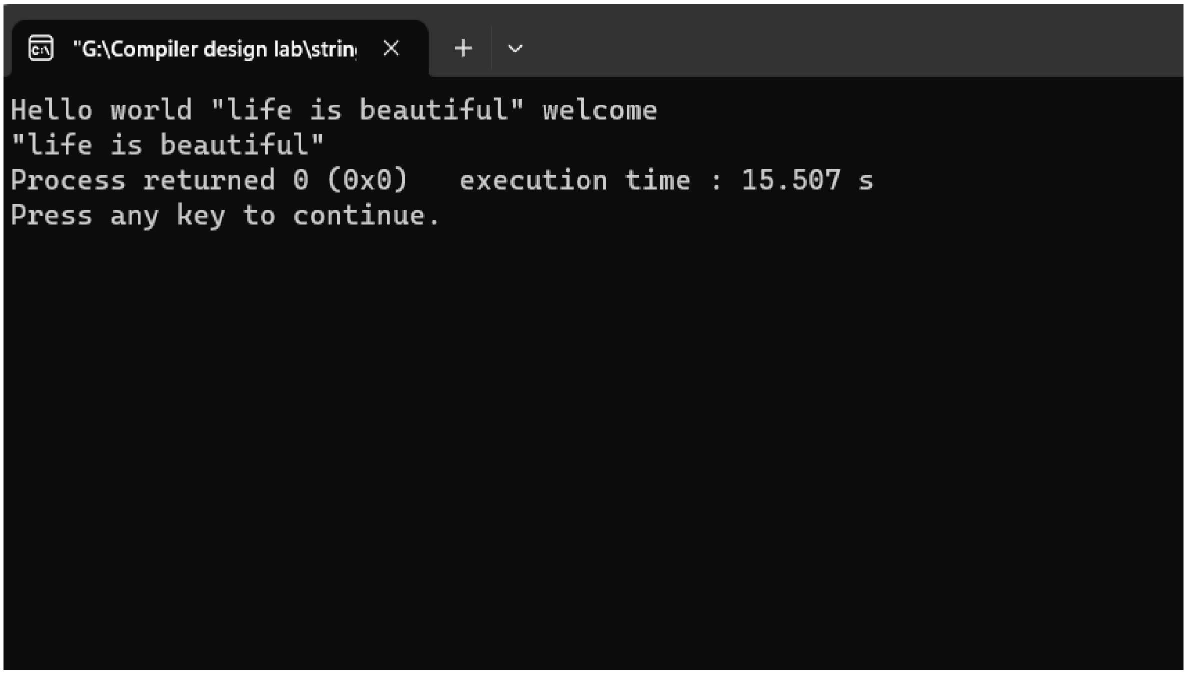
A string is a sequence of characters and can contain letters, numbers, symbols and even spaces. It must be enclosed in quotation marks for it to be recognized as a string.

Code:

```
#include<stdio.h>

#include<string.h>

int main(){


char str[50];


gets(str);


int len = strlen(str);

int i,j =0,k,c =0, arr[len];


for(i=0;i<len; ++i){


   if(str[i] =='"'){
        c++;

        arr[j]= i;

        j++;

    }

}


if(c ==0){

    printf("There is no string");

}else if(c == 2){


for(k =arr[0];k<=arr[1]; k++){

    printf("%c",str[k]);

}

}

else{

 printf("There is an compiler error");


}
```

```
return 0;

}
```

Output:



```
Hello world "life is beautiful" welcome
"life is beautiful"
Process returned 0 (0x0)   execution time : 15.507 s
Press any key to continue.
```

Problem 04:

Objective: Write a Program to detect number of '/' in a string.

Description:

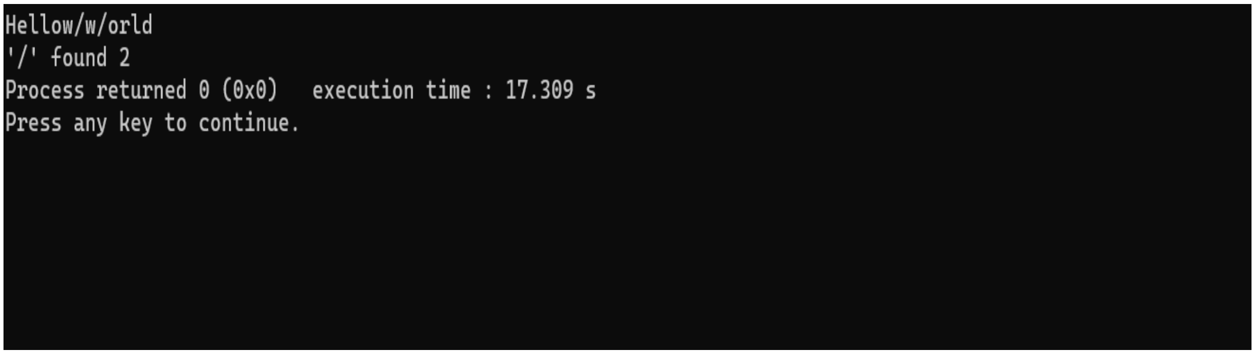
Sample Input: Hellow/w/orld

Sample Output: '/' found 2

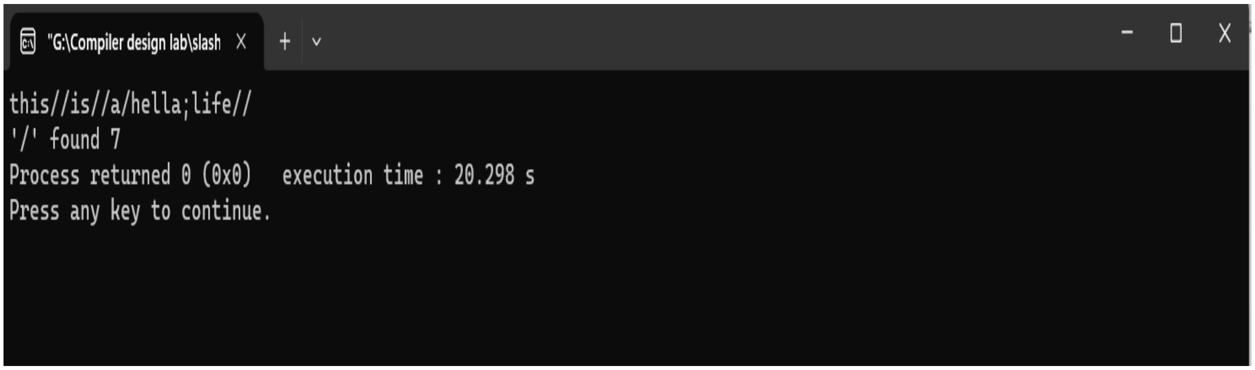
Code:

```
#include<stdio.h>


int main(){


char str[50];


int c=0;


gets(str);


int len = strlen(str);


for(int i =0; i< len;++i){


    if(str[i] == '/'){
        c++;
    }
}
 printf("'/' found %d",c);
 return 0;
}
```

<u>Output:</u>



```
Hellow/w/orld
'/' found 2
Process returned 0 (0x0)   execution time : 17.309 s
Press any key to continue.
```



```
this//is//a/hella;life//
'/' found 7
Process returned 0 (0x0)   execution time : 20.298 s
Press any key to continue.
```

Problem 05:

Objective: Design a Lexical Analyzer to detect the following tokens.

1. Keywords [All 32].
2. Identifiers.
3. Operators.

Description:

Sample Input:

int sum = a + b;

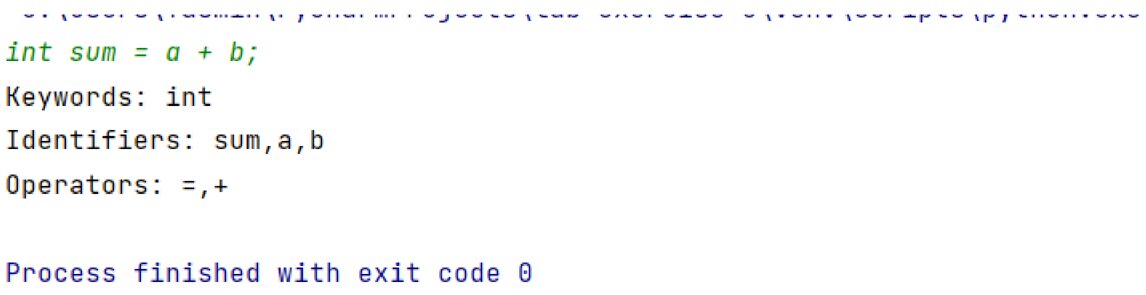
Sample Output:

Keywords: int.
Identifiers: sum, a, b.
Operators: =, +.

Code:

```
keywords = ['auto', 'break', 'case', 'const', 'continue', 'default', 'do',
'double',
            'else', 'enum', 'extern', 'float', 'for', 'goto', 'if', 'int',
'long', 'register',
            'return', 'short', 'signed', 'sizeof', 'static', 'struct',
'switch', 'typedef', 'union',
            'unsigned', 'void', 'volatile', 'while']
operator = ['+', '-', '%', '&', '=']
op = []
kw = []
tokens = []

from string import ascii_lowercase, ascii_uppercase

ids = list(ascii_lowercase)
ids_u = list(ascii_uppercase)
identifiers = []
import re

line = input("")
for i in line:
    if i in operator:
        op.append(i)
tokens = re.split('\\s|(?<!\\d)[-+%&=,](?!\\d)|;', line)

for token in tokens:
    if token in keywords:
        kw.append(token)
    elif token in ids:
        identifiers.append(token)
    elif token in ids_u:
        identifiers.append(token)
    elif token.isidentifier():
        identifiers.append(token)

print("Keywords:", ','.join(map(str, kw)))
print("Identifiers:", ','.join(map(str, identifiers)))
print("Operators:", ','.join(map(str, op)))
```

Output:



```
int sum = a + b;
Keywords: int
Identifiers: sum,a,b
Operators: =,+

Process finished with exit code 0
```

Problem 06:

Description: Write a code to detect valid identifier or not.

In C, a valid identifier is a name used to identify a variable, function, or other entity in the program. The rules for valid identifiers in C are as follows:

- An identifier must start with a letter (a-z, A-Z) or an underscore (_). It cannot start with a digit (0-9).

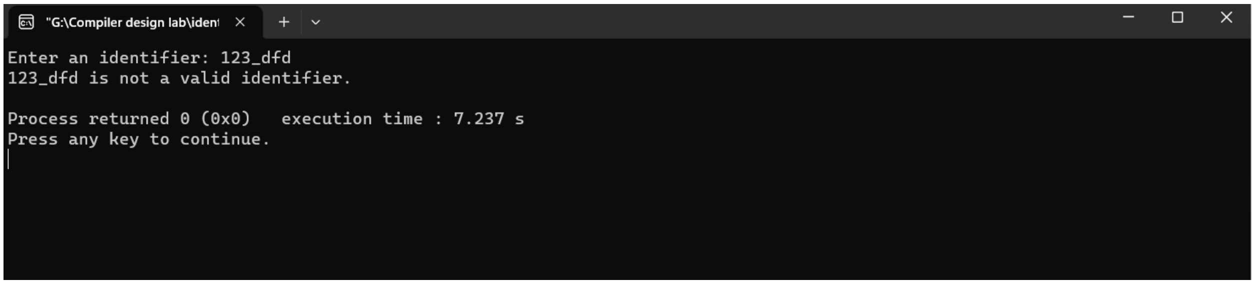
- After the initial character, an identifier can contain letters, digits, and underscores.
- Identifiers are case-sensitive. For example, myVariable and myvariable are treated as different identifiers.
- The maximum length of an identifier may vary but is typically limited to 31 or 63 characters in most compilers.

Code:

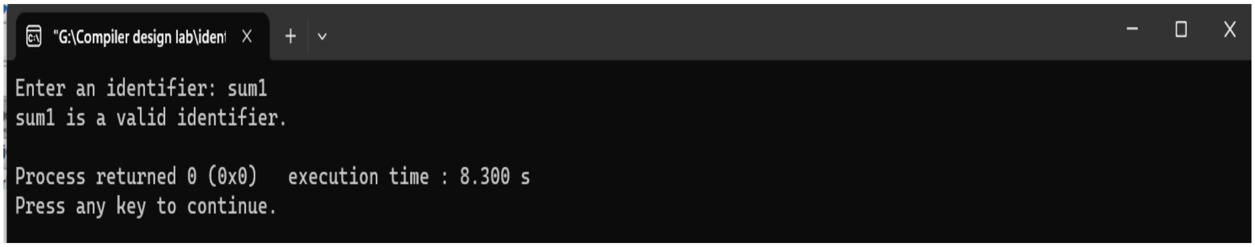
```
#include <stdio.h>

#include <string.h>


int isValidIdentifier( char*  identifier) {

  if (!((identifier[0] >= 'a' && identifier[0] <= 'z') ||

      (identifier[0] >= 'A' && identifier[0] <= 'Z') ||

      identifier[0] == '_')) {

    return 0;

  }



  for (int i = 1; i < strlen(identifier); i++) {
    if (!((identifier[i] >= 'a' && identifier[i] <= 'z') ||
          (identifier[i] >= 'A' && identifier[i] <= 'Z') ||
          (identifier[i] >= '0' && identifier[i] <= '9') ||
          identifier[i] == '_')) {
        return 0;
    }
  }

  return 1;
}

int main() {
  char identifier[50];
  printf("Enter an identifier: ");
  scanf("%s", identifier);

  if (isValidIdentifier(identifier)) {
    printf("%s is a valid identifier.\n", identifier);
```

```
  } else {
    printf("%s is not a valid identifier.\n", identifier);
  }

  return 0;
}
```

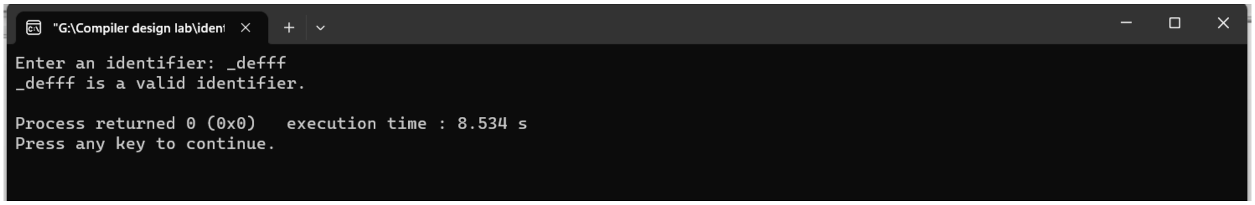
Output:



```
Enter an identifier: 123_dfd
123_dfd is not a valid identifier.

Process returned 0 (0x0)   execution time : 7.237 s
Press any key to continue.
```



```
Enter an identifier: sum1
sum1 is a valid identifier.

Process returned 0 (0x0)   execution time : 8.300 s
Press any key to continue.
```



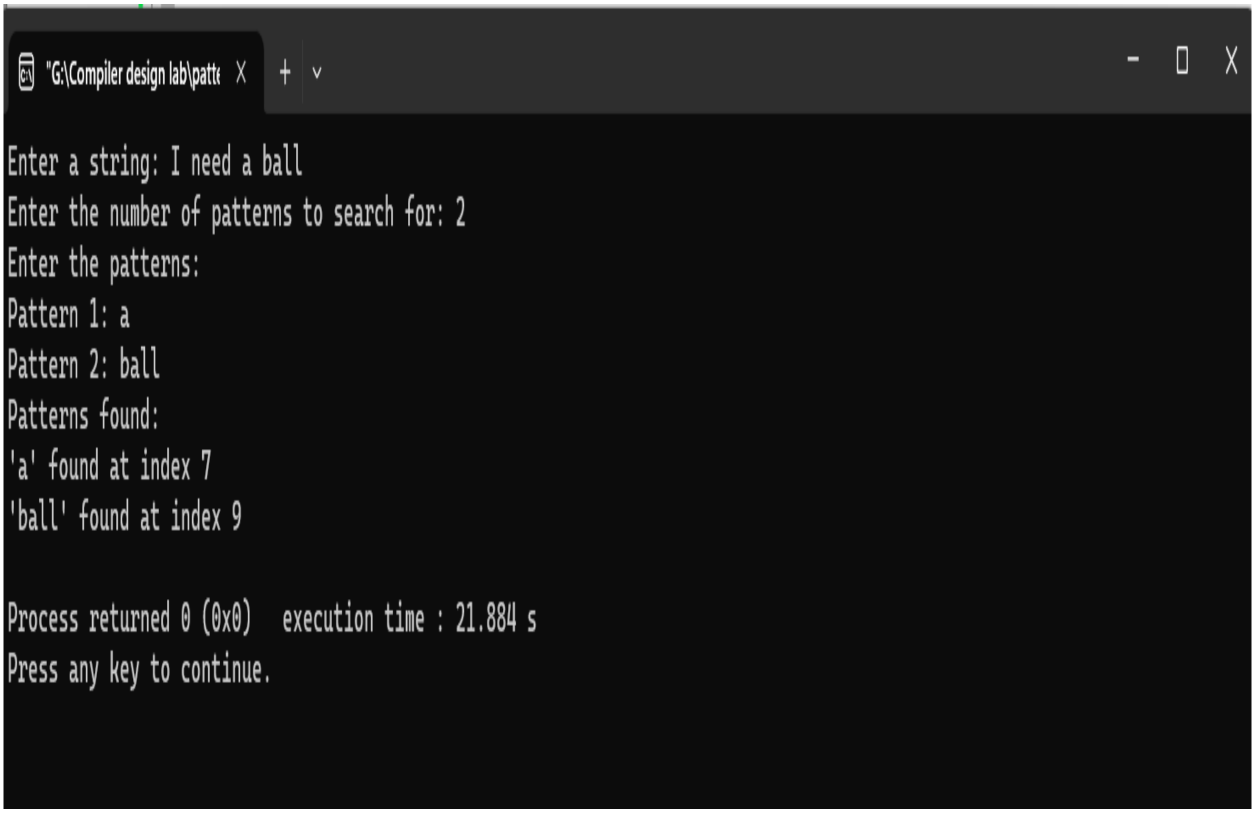
```
Enter an identifier: _defff
_defff is a valid identifier.

Process returned 0 (0x0)   execution time : 8.534 s
Press any key to continue.
```

Problem 7:

Code:

```
#include <stdio.h>

#include <string.h>


int findPattern(const char *str, const char *pattern) {

    int len = strlen(str);

    int patLen = strlen(pattern);


    for (int i = 0; i <= len - patLen; i++) {

        int j;

        for (j = 0; j < patLen; j++) {

            if (str[i + j] != pattern[j])

                break;

        }

        if (j == patLen)

            return i;  // Pattern found at index i

    }
    return -1;  /
}

int main() {
    char str[100];
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';

    int numPatterns;
    printf("Enter the number of patterns to search for: ");
    scanf("%d", &numPatterns);

    char patterns[numPatterns][100];
    getchar();

    printf("Enter the patterns:\n");
    for (int i = 0; i < numPatterns; i++) {
        printf("Pattern %d: ", i + 1);
        fgets(patterns[i], sizeof(patterns[i]), stdin);
        patterns[i][strcspn(patterns[i], "\n")] = '\0';
    }

    printf("Patterns found:\n");
    for (int i = 0; i < numPatterns; i++) {
        int index = findPattern(str, patterns[i]);
        if (index != -1) {
            printf("'%s' found at index %d\n", patterns[i], index);
        }
    }

    return 0;
}
```

Output:



```
Enter a string: I need a ball
Enter the number of patterns to search for: 2
Enter the patterns:
Pattern 1: a
Pattern 2: ball
Patterns found:
'a' found at index 7
'ball' found at index 9

Process returned 0 (0x0)   execution time : 21.884 s
Press any key to continue.
```