Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Index File

Student Name	: Md Khaled Bin Joha
Student ID	: 0822220105101052
Name of the Course	: Compiler Design and Construction Sessional
Course Code	: CSE-414

Lab Report No	Lab Report Name	Submission Date	Total Marks	Obtained Marks
1	Write a Program to Test Whether an Identifier is Valid or Not.	08/01/2026	10	
2	Write a Program to Identify Whether a Given Line is a Comment or Not.	08/01/2026	10	
3	Write a Program to Scan and Count the Number of Characters, Words and Lines in a File.	08/01/2026	10	
4	Write a Program to Check Whether a Given String is a Keyword or Not. Using file I/O	08/01/2026	10	
5	Write a Program to Check Whether a Given String is a Constant or Not.	08/01/2026	10	
6	Write a Program to Count the Blank Spaces in a String	08/01/2026	10	
7	Write a Program to Find FIRST in a Context Free Grammar	08/01/2026	10	
8	Write a Program to Detect Tokens in a CP Program	08/01/2026	10	
9	A simple calculator that evaluates basic arithmetic expressions (addition and subtraction) using Lex for tokenization and YACC	08/01/2026	10	
10			10	

Total marks obtained :

Total number of lab reports :

Average marks :

Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	1						
Lab Report Name	:	Write a Program to Test Whether an Identifier is Valid or Not.						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- Understand the rules for valid identifiers in programming languages like C (e.g., starts with letter or underscore, followed by letters, digits, or underscores; not a keyword).
- Learn to implement string validation logic using character checks.

Code Implementation:

Lab Works > lab1.c > ...

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

char keywords[32][10] ={
    "auto","break","case","char","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"
};

int isKeyword(char *str){
    for(int i = 0; i < 32; i++){
        if (strcmp(str, keywords[i]) == 0)
            return 1;
    }
    return 0;
}

int isValidIdentifier(char *str){
    if(!(isalpha(str[0]) || str[0] == '_'))
        return 0;

    for(int i = 1; str[i] != '\0'; i++){
        if(!(isalnum(str[i]) || str[i] == '_')){
            return 0;
        }
```

```c
int isValidIdentifier(char *str){

    for(int i = 1; str[i] != '\0'; i++){
        if(!(isalnum(str[i]) || str[i] == '_')){
            return 0;
        }

    }

    if(isKeyword(str))
        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 an INVALID identifier.\n", identifier);

    return 0;
}
```

Input Sample:
Enter an identifier: sum_1

Output Sample:

```
joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ ./lab1
Enter an identifier: sum_1
'sum_1' is a VALID identifier.
```

Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	2						
Lab Report Name	:	Write a Program to Identify Whether a Given Line is a Comment or Not.						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- Implement string pattern matching to detect comment starters.
- Handle edge cases such as nested comments or strings containing comment-like patterns.

Code Implementation:

Lab Works > lab2.c > main()

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

bool isComment(char *line){
    int i = 0;
    int len = strlen(line);
    bool inString = false;
    bool inChar = false;
    bool escapeNext = false;

    while(i < len){
        if(escapeNext){
            escapeNext = false;
            i++;
            continue;
        }

        if(line[i] == '\\'){
            escapeNext = true;
            i++;
            continue;
        }

        // handling string literals
        if(line[i] == '"' && !inChar){
            inString = !inString;
            i++;
```

Lab Works > C lab2.c > main()

```
 5  bool isComment(char *line){
12      while(i < len){
24
25          // handling string literals
26          if(line[i] == '"' && !inChar){
27              inString = !inString;
28              i++;
29              continue;
30          }
31
32          // handle character literals
33          if(line[i] == '\'' && !inString){
34              inChar = !inChar;
35              i++;
36              continue;
37          }
38
39          if(!inString && !inChar){
40              if(i < len-1 && line[i] == '/' && line[i+1] =
41                  return true;
42              }
43          }
44          i++;
45      }
46      return false;
47  }
48
49  int main(){
```

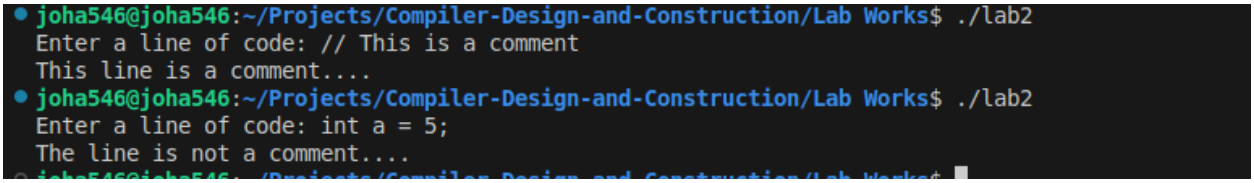
```
48
49  int main(){
50      char line[1000];
51
52      printf("Enter a line of code: ");
53      fgets(line, sizeof(line), stdin);
54
55      line[strcspn(line, "\n")] = '\0';
56
57      if(isComment(line)){
58          printf("This line is a comment....\n");
59      }
60      else{
61          printf("The line is not a comment....\n");
62      }
63      return 0;
64  }
65
```

Input Sample:

Enter a line of code: // This is a comment

Enter a line of code: int a = 5;

Output Sample:



Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	3						
Lab Report Name	:	Write a Program to Scan and Count the Number of Characters, Words and Lines in a File.						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

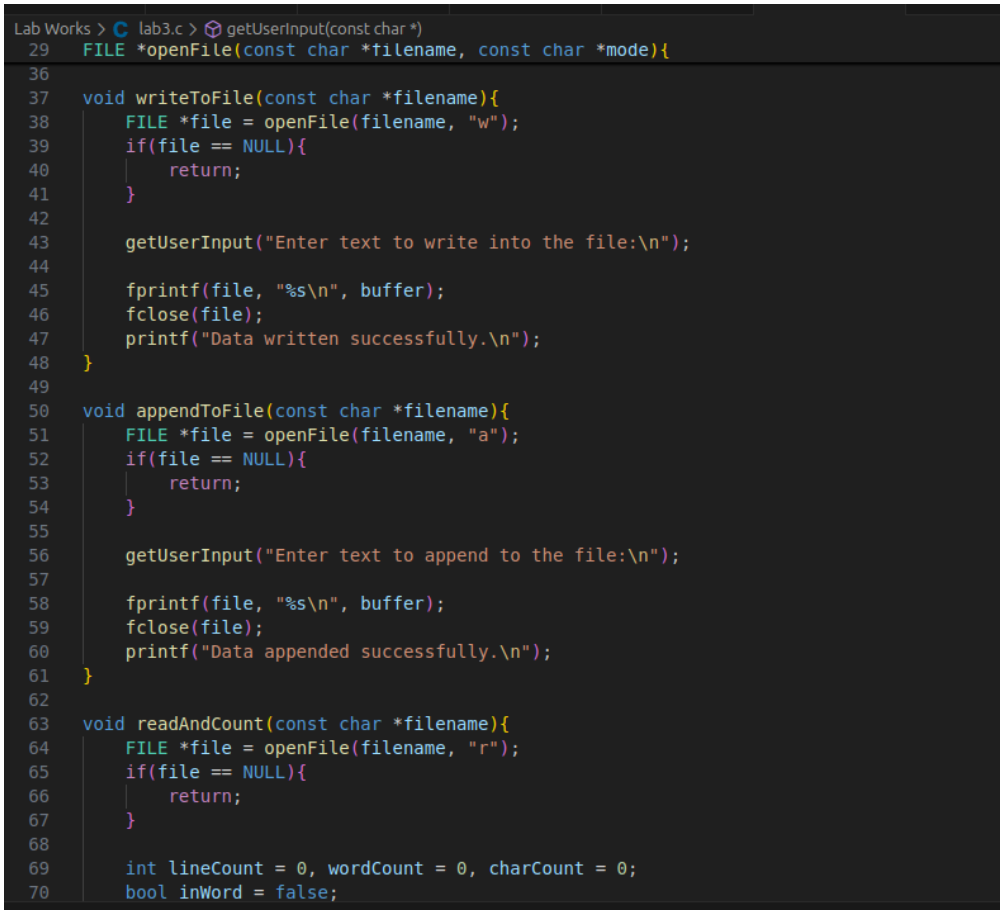
Key Learnings:

- Learn file input/output operations in C.
- Implement counters for characters, words (space-separated), and lines (newline-separated).
- Handle edge cases like empty files or trailing spaces.

Code Implementation:

Lab Works > C lab3.c > getUserInput(const char *)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>

#define BUFFER_SIZE 4096
#define MAX_FILENAME_LENGTH 255

char buffer[BUFFER_SIZE];

void clearInputBuffer(){
    int c;
    while((c = getchar()) != '\n' && c != EOF);
}

void getUserInput(const char *prompt){
    printf("%s", prompt);
    clearInputBuffer();

    if(fgets(buffer, BUFFER_SIZE, stdin) != NULL){
        size_t len = strlen(buffer);
        if(len > 0 && buffer[len - 1] == '\n'){
            buffer[len - 1] = '\0';
        }
    }
}

FILE *openFile(const char *filename, const char *mode){
    FILE *file = fopen(filename, mode);
    if(file == NULL){
        perror("Error opening file");
    }
    return file;
}
```


Lab Works > lab3.c > getUserInput(const char *)
29 FILE *openFile(const char *filename, const char *mode){
36
37 void writeToFile(const char *filename){
38 FILE *file = openFile(filename, "w");
39 if(file == NULL){
40 return;
41 }
42
43 getUserInput("Enter text to write into the file:\n");
44
45 fprintf(file, "%s\n", buffer);
46 fclose(file);
47 printf("Data written successfully.\n");
48 }
49
50 void appendToFile(const char *filename){
51 FILE *file = openFile(filename, "a");
52 if(file == NULL){
53 return;
54 }
55
56 getUserInput("Enter text to append to the file:\n");
57
58 fprintf(file, "%s\n", buffer);
59 fclose(file);
60 printf("Data appended successfully.\n");
61 }
62
63 void readAndCount(const char *filename){
64 FILE *file = openFile(filename, "r");
65 if(file == NULL){
66 return;
67 }
68
69 int lineCount = 0, wordCount = 0, charCount = 0;
70 bool inWord = false;

Lab Works > lab3.c > getUserInput(const char *)

```c
void readAndCount(const char *filename){
    bool inWord = false;

    printf("\n--- File Content ---\n");

    while(fgets(buffer, BUFFER_SIZE, file) != NULL){
        printf("%s", buffer);
        lineCount++;

        size_t len = strlen(buffer);
        // Not to count new line character
        if(len > 0 && buffer[len - 1] == '\n'){
            charCount += len - 1;
        }
        else{
            charCount += len;
        }

        for(size_t i = 0; i < len; i++){
            if(isspace((unsigned char)buffer[i])){
                inWord = false;
            }
            else if(!inWord){
                inWord = true;
                wordCount++;
            }
        }
    }

    fclose(file);

    printf("\n--- File Statistics ---\n");
    printf("Lines: %d\n", lineCount);
    printf("Words: %d\n", wordCount);
    printf("Characters: %d\n", charCount);
}
```

Lab Works > C lab3.c > getUserInput(const char *)

```c
int main() {
    char filename[MAX_FILENAME_LENGTH + 1];
    int choice;

    printf("Enter file name: ");
    if(scanf("%255s", filename) != 1){
        printf("Error reading filename.\n");
        return 1;
    }

    while(1){
        printf("\n--- File Operations Menu ---\n");
        printf("1. Write to file\n");
        printf("2. Append to file\n");
        printf("3. Read & Count file\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");

        if(scanf("%d", &choice) != 1){
            printf("Invalid input. Please enter a number.\n");
            clearInputBuffer();
            continue;
        }

        switch(choice){
            case 1:
                writeToFile(filename);
                break;
            case 2:
                appendToFile(filename);
                break;
            case 3:
                readAndCount(filename);
                break;
            case 4:
                printf("Exiting program...\n");
```

```c
                break;
            case 4:
                printf("Exiting program...\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
}
```

Input Sample:

./lab3

Enter file name: data.txt

--- File Operations Menu ---

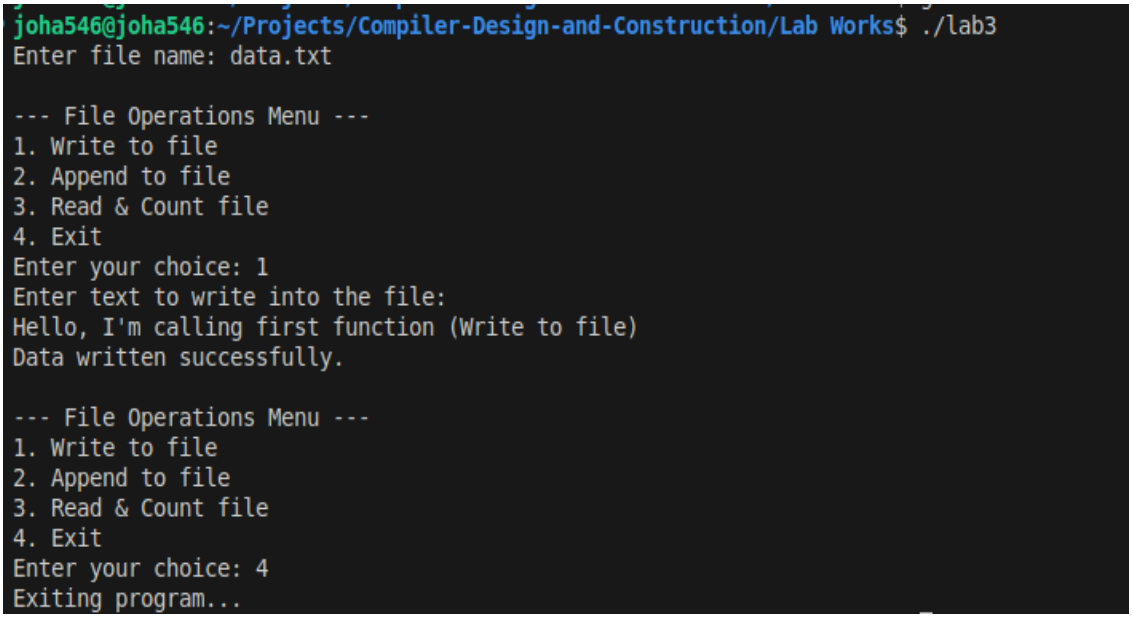
1. Write to file
2. Append to file
3. Read & Count file
4. Exit

Enter your choice: 1

Enter text to write into the file:

Hello, I'm calling first function (Write to file)

Output Sample:



Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	4						
Lab Report Name	:	Write a Program to Check Whether a Given String is a Keyword or Not. Using file I/O						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- **Understanding File I/O in C:** Learned how to read strings from a file and process them programmatically using file handling functions (fopen, fscanf, fgets, fclose).
- **Keyword Identification Logic:** Gained clarity on how compilers distinguish **keywords from identifiers** by comparing input strings against a predefined keyword list.
- **Practical Insight into Lexical Analysis:** Understood how keyword checking is a fundamental part of the **lexical analysis phase** in compiler design.

Code Implementation:

Lab Works > lab4.c > readKeywords(char [][MAX_KEYWORD_LENGTH], const char *)

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

#define MAX_KEYWORDS 50
#define MAX_KEYWORD_LENGTH 20

int readKeywords(char keywords[][MAX_KEYWORD_LENGTH], const char* filename){
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        printf("Error opening keywords file.\n");
        return 0;
    }

    int count = 0;
    while (fscanf(file, "%s", keywords[count]) == 1 && count < MAX_KEYWORDS)
        count++;
    }
    printf("%d", count);
    fclose(file);
    return count;
}

int isKeyword(char str[], char keywords[][MAX_KEYWORD_LENGTH], int keywordCou
    for(int i=0; i<keywordCount; i++){
        if(strcmp(str, keywords[i]) == 0){
            return 1;
        }
    }
    return 0;
}

```

```c
int main(){
    char str[MAX_KEYWORD_LENGTH];
    char keywords[MAX_KEYWORDS][MAX_KEYWORD_LENGTH];
    char keywordFilename[40];
    int keywordCount;

    scanf("%s", keywordFilename);

    keywordCount = readKeywords(keywords, keywordFilename);
    if(keywordCount == 0){
        printf("No keywords loaded. Exiting.\n");
        return 1;
    }

    while(getchar() != '\n');

    scanf("%s", str);

    if(isKeyword(str, keywords, keywordCount)){
        printf("\"%s\" is a keyword.\n", str);
    }
    else{
        printf("\"%s\" is not a keyword.\n", str);
    }
    return 0;
}
```

Input Sample:
textfile.txt
32
auto
Another:
textfile.txt
32
notAKeyword

Output Sample:

```
joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ ./lab4
textfile.txt
32
auto
"auto" is a keyword.
joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ ./lab4
textfile.txt
32
notAKeyword
"notAKeyword" is not a keyword.
```

Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	5						
Lab Report Name	:	Write a Program to Check Whether a Given String is a Constant or Not.						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- Identify types of constants (integer, float, character, string).
- Use character checks to validate numeric or literal formats.
- Differentiate constants from identifiers or other tokens.

Code Implementation:

Lab Works > lab5.c > isFloat(char [])

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

int isInteger(char str[]){
    int i = 0;
    if(str[i] == '+' || str[i] == '-')
        i++;
    if(!isdigit(str[i]))
        return 0;
    for(; str[i] != '\0'; i++){
        if(!isdigit(str[i]))
            return 0;
    }
    return 1;
}

int isFloat(char str[]){
    int i = 0, dotCount = 0, digitCount = 0;
    if(str[i] == '+' || str[i] == '-'){
        i++;
    }
    for(; str[i] != '\0'; i++){
        if(str[i] == '.'){
            dotCount++;
            if(dotCount > 1)
                return 0;
        }
        else if(isdigit(str[i])){
            digitCount++;
        }
        else{
            return 0;
        }
    }
    return (dotCount == 1 && digitCount > 0);
}
```

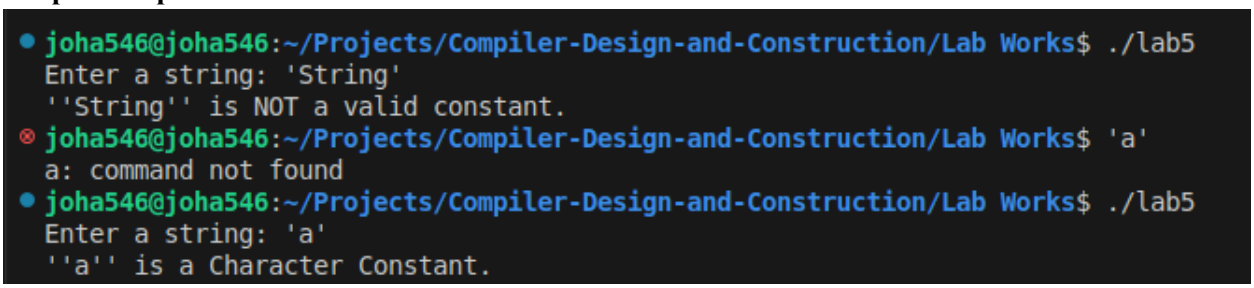
```c
int isCharacterConstant(char str[]) {
    int len = strlen(str);
    if(len == 3 && str[0] == '\'' && str[2] == '\''){
        return 1;
    }
    if(len == 4 && str[0] == '\'' && str[1] == '\\' && str[3] == '\'')
        return 1;
    return 0;
}

int isStringConstant(char str[]) {
    int len = strlen(str);
    return (len >= 2 && str[0] == '"' && str[len - 1] == '"');
}

int main() {
    char str[300];
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';

    if(isInteger(str)){
        printf("'%s' is an Integer Constant.\n", str);
    }
    else if(isFloat(str)){
        printf("'%s' is a Floating-Point Constant.\n", str);
    }
    else if(isCharacterConstant(str)){
        printf("'%s' is a Character Constant.\n", str);
    }
    else if(isStringConstant(str)){
        printf("'%s' is a String Constant.\n", str);
    }
    else{
        printf("'%s' is NOT a valid constant.\n", str);
    }
    return 0;
}
```

Input Sample:

Enter a string: 'String'

Enter a string: 'a'

Output Sample:



Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	6						
Lab Report Name	:	Write a Program to Count the Blank Spaces in a String						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- Practice string traversal and character inspection.
- Use isspace() to detect various whitespace (space, tab, etc.).
- Handle input strings with leading/trailing spaces

Code Implementation:

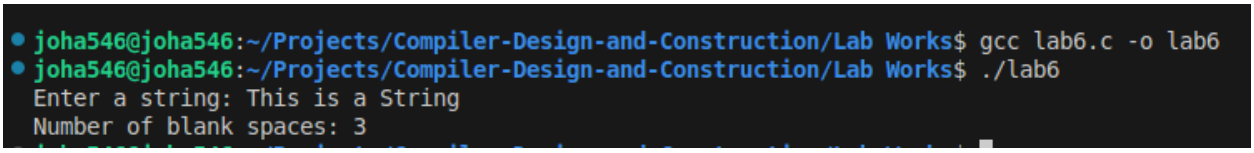
Lab Works > lab6.c > main()

```c
#include <stdio.h>

int main(){
    char str[200];
    int count = 0, i;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    for(i = 0; str[i] != '\0'; i++){
        if(str[i] == ' ')
            count++;
    }

    printf("Number of blank spaces: %d\n", count);

    return 0;
}
```

Input Sample:

Enter a string: This is a String

Output Sample:


joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ gcc lab6.c -o lab6
joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ ./lab6
Enter a string: This is a String
Number of blank spaces: 3

Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	7						
Lab Report Name	:	Write a Program to Find FIRST in a Context Free Grammar						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- **Understanding FIRST Sets in CFGs:** Learned how to compute the FIRST set for terminals and non-terminals, which is a fundamental concept in **syntax analysis** and **parsing**.
- **Recursive Algorithm Implementation:** Gained experience in designing **recursive functions** to handle grammar rules and non-terminal dependencies**Practical Compiler Insight:** Developed insight into how **lexical tokens and grammar rules** are analyzed by a compiler to predict possible starting symbols of productions.

Code Implementation:

Lab Works > lab7.c > main()

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

int n;
char production[10][10];
char first[10][10];
int firstCount[10];

void addToFirst(int index, char ch){
    for(int i=0; i<firstCount[index]; i++){
        if(first[index][i] == ch){
            return;
        }
    }

    first[index][firstCount[index]++] = ch;
}

void findFirst(int index){
    char rhs[10];
    strcpy(rhs, production[index]+2);

    if(!isupper(rhs[0])){
        addToFirst(index, rhs[0]);
        return;
    }

    int ntIndex = rhs[0] - 'A';
    findFirst(ntIndex);
}
```

```c

int main(){
    printf("Enter number of productions: ");
    scanf("%d", &n);

    printf("Enter productions {For example: E=TR:}\n");
    for(int i=0; i<n; i++){
        scanf("%s", production[i]);
        firstCount[i] = 0;
    }

    for(int i=0; i<n; i++){
        findFirst(i);
    }

    printf("\nFirst sets: \n");
    for(int i=0; i<n; i++){
        printf("First(%c) = { ", production[i][0]);
        for(int j=0; j<firstCount[i]; j++){
            printf("%c", first[i][j]);
        }
        printf("}\n");
    }

    return 0;
}
```

Input Sample:

Enter number of productions: 3

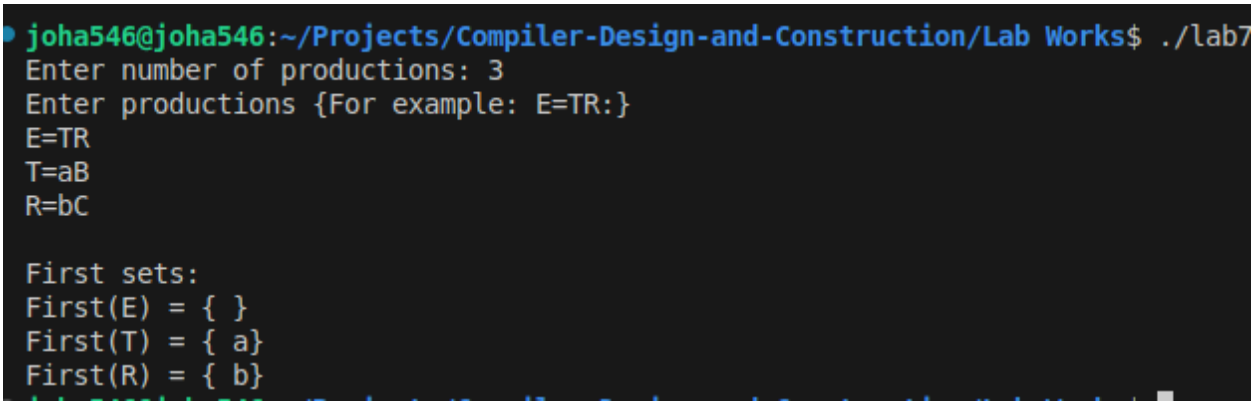
Enter productions {For example: E=TR:}

E=TR

T=aB

R=bC

Output Sample:


joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ ./lab7
Enter number of productions: 3
Enter productions {For example: E=TR:}
E=TR
T=aB
R=bC
First sets:
First(E) = { }
First(T) = { a}
First(R) = { b}

Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	8						
Lab Report Name	:	Write a Program to Detect Tokens in a CP Program						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- **Lexical Analysis Fundamentals:** Learned how to break a source code file into **tokens** such as keywords, identifiers, numbers, operators, and special symbols.
- **File I/O in C:** Practiced reading source code from a file character by character using fgetc() and processing it programmatically.
- **Practical Compiler Insight:** Gained hands-on understanding of how **compilers or interpreters recognize language elements** before parsing.

Code Implementation:

Lab Works > lab8.c > main()

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

char keywords[10][10] = {
    "int", "float", "if", "else", "while",
    "for", "return", "char", "double", "void"
};

int isKeyword(char *word){
    for(int i=0; i<10; i++){
        if(strcmp(keywords[i], word) == 0){
            return 1;
        }
    }
    return 0;
}

int main(){
    char ch, buffer[50];
    int i=0;

    FILE *fp = fopen("input.c", "r");

    if(fp == NULL){
        printf("Error opening file\n");
        return 0;
    }

    printf("TOKENS FOUND\n");

    while((ch = fgetc(fp)) != EOF){
        // Preprocessor Symbol
        if(ch == '#'){
            printf("# -> Special symbol\n");

            fscanf(fp, "%s", buffer);
            printf("%s -> Preprocessor Directive\n", buffer);

```

Lab Works > lab8.c > main()

```c
            fgetc(fp);
            printf("< -> Special symbol\n");

            fscanf(fp, "%[^>]", buffer);
            printf("%s -> Header File\n", buffer);

            fgetc(fp);
            printf("> -> Special symbol\n");
        }
        else if(isalpha(ch)){
            buffer[i++] = ch;
            while(isalnum(ch = fgetc(fp))){
                buffer[i++] = ch;
            }

            buffer[i] = '\0';

            if(isKeyword(buffer)){
                printf("%s -> Keyword\n", buffer);
            }
            else{
                printf("%s -> Identifier\n", buffer);
            }

            i = 0;
            ungetc(ch, fp);
        }
        else if(isdigit(ch)){
            buffer[i++] = ch;
            while(isdigit(ch = fgetc(fp))){
                buffer[i++] = ch;
            }

            buffer[i] = '\0';
            printf("%s -> Number\n", buffer);

            i = 0;
            ungetc(ch, fp);
        }

```

```
79
80          else if(ch == '+' || ch == '-' ||  ch == '*' || ch == '/' || ch == '<' || ch == '>'){
81              printf("%c -> Operator\n", ch);
82          }
83
84          else if(ch == ';' || ch == ',' || ch == '(' || ch == ')' || ch == '{' || ch == '}'){
85              printf("%c -> Special Symbol\n", ch);
86          }
87      }
88      fclose(fp);
89      return 0;
90  }
```

Output Sample:

```
joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works$ ./lab8
TOKENS FOUND
# -> Special symbol
include -> Preprocessor Directive
< -> Special symbol
<stdio.h -> Header File
> -> Special symbol
int -> Keyword
main -> Identifier
( -> Special Symbol
) -> Special Symbol
{ -> Special Symbol
int -> Keyword
a -> Identifier
10 -> Number
; -> Special Symbol
float -> Keyword
b -> Identifier
a -> Identifier
+ -> Operator
5 -> Number
; -> Special Symbol
return -> Keyword
0 -> Number
; -> Special Symbol
} -> Special Symbol
```

Bangladesh Army International University of Science and Technology

Department of Computer Science and Engineering

Lab Report

Lab Report No	:	9						
Lab Report Name	:	A simple calculator that evaluates basic arithmetic expressions (addition and subtraction) using Lex for tokenization and YACC						
Course Title	:	Compiler Design and Construction Sessional						
Course Code	:	CSE-414						
Name	:	Md Khaled Bin Joha						
ID	:	0822220105101052						
Level	:	4	**Term**	: 1	**Section**	: B	**Group**	:
Date of Submission	:	08/01/26	**Semester**	: FALL 25	**Year**	: 2026		

Key Learnings:

- Understand the role of lexical analysis (tokenization)
- Understand the role of syntax analysis (parsing)
- Write Lex specifications for tokens
- Write YACC grammar rules
- Build simple language processors
- Debug compilation errors
- Trace expression evaluation

Code Implementation:

Lab Works > lab9 > calc.l

```
%{
#include "y.tab.h"
%}

%%
[0-9]+      { yylval = atoi(yytext); return NUMBER; }
[+]         { return PLUS; }
[-]         { return MINUS; }
[\n]        { return 0; }
[ \t]       { /* ignore whitespace */ }
.           { printf("Invalid character: %s\n", yytext); }
%%
```

Lab Works > lab9 > calc.y

```
%{
#include <stdio.h>
#include <stdlib.h>
int yylex();
void yyerror(char *s);
%}

%token NUMBER PLUS MINUS

%%
expression:
      expression PLUS expression   { $$ = $1 + $3; printf("Result = %d\n", $$); }
    | expression MINUS expression  { $$ = $1 - $3; printf("Result = %d\n", $$); }
    | NUMBER                       { $$ = $1; }
    ;
%%
void yyerror(char *s) {
    printf("Error: %s\n", s);
}

int main() {
    printf("Enter arithmetic expression:\n");
    yyparse();
    return 0;
}
```

Input Sample:

Enter arithmetic expression:

5+8

Output Sample:

```
joha546@joha546:~/Projects/Compiler-Design-and-Construction/Lab Works/lab9$ ./calc
Enter arithmetic expression:
5+8
Result = 13
```