BCSE307P
Compiler Design Lab
Lab Assignment 1

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

Name – Ishan Kapoor
Registration Number – 21BCE5882
Submitted to – Prof. S. Srisakthi

1. Write a C program to identify whether a given line is a comment or not.

CODE:

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

bool isComment(const char *line) {
    // Check for single-line comment
    if (strncmp(line, "//", 2) == 0) {
        return true;
    }

    // Check for multi-line comment
    if (strncmp(line, "/*", 2) == 0) {
        return true;
    }

    return false;
}
int main() {
    char line[100];

    // Read the line from input
    printf("Enter a line: ");
    fgets(line, sizeof(line), stdin);

    // Remove the newline character at the end
    line[strcspn(line, "\n")] = '\0';

    // Check if the line is a comment
    if (isComment(line)) {
        printf("The line is a comment.\n");
    } else {
        printf("The line is not a comment.\n");
    }

    return 0;
}
```

OUTPUT:

```
Enter a line: //x = x+1;
The line is a comment.

--------------------------------
Process exited after 68.78 seconds with return value 0
Press any key to continue . . .
```

```
Enter a line: /*x=x+1*/
The line is a comment.

--------------------------------
Process exited after 8.642 seconds with return value 0
Press any key to continue . . .
```

```
Enter a line: x = x+1;
The line is not a comment.

--------------------------------
Process exited after 5.961 seconds with return value 0
Press any key to continue . . .
```

2. Write a C program to recognize strings under ‘a’, ‘a*b+’, ‘abb’.

CODE:

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
char s[20],c;
int state=0,i=0;
printf("\n Enter a string: ");
gets(s);
while(s[i]!='\0')
{
switch(state)
{
case 0: c=s[i++];
if(c=='a')
state=1;
else if(c=='b')
state=2;
else
state=6;
break;
```

```
case 1: c=s[i++];
if(c=='a')
state=3;
else if(c=='b')
state=4;
else
state=6;
break;
case 2: c=s[i++];
if(c=='a')
state=6;
else if(c=='b')
state=2;
else
state=6;
break;
case 3: c=s[i++];
if(c=='a')
state=3;
else if(c=='b')
state=2;
else
state=6;
break;
case 4: c=s[i++];
if(c=='a')
state=6;
else if(c=='b')
state=5;
else
state=6;
break;
case 5: c=s[i++];
if(c=='a')
```

```
state=6;
else if(c=='b')
state=2;
else
state=6;
break;
case 6: printf("\n %s is not recognised.",s);
exit(0);
}
}
if(state==1)
printf("\n %s is accepted under rule 'a'",s);
else if((state==2)||(state==4))
printf("\n %s is accepted under rule 'a*b+'",s);
else if(state==5)
printf("\n %s is accepted under rule 'abb'",s);
getch();
}
```

OUTPUT:

```
 Enter a string: ab

 ab is accepted under rule 'a*b+'
--------------------------------
Process exited after 7.618 seconds with return value 13
Press any key to continue . . .
```

```
 Enter a string: aab

 aab is accepted under rule 'a*b+'
--------------------------------
Process exited after 5.32 seconds with return value 13
Press any key to continue . . .
```

```
 Enter a string: abb

 abb is accepted under rule 'abb'
--------------------------------
Process exited after 9.228 seconds with return value 13
Press any key to continue . . .
```

```
 Enter a string: bab

 bab is not recognised.
--------------------------------
Process exited after 1.533 seconds with return value 0
Press any key to continue . . .
```

3. Tokenizing a file using C.

CODE:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX_TOKEN_LENGTH 100

// Function to check if a character is a valid identifier character
int isValidIdentifierChar(char ch) {
    return isalnum(ch) || ch == '_';
}
// Function to tokenize a string
void tokenizeString(const char* input) {
    int length = strlen(input);
    char token[MAX_TOKEN_LENGTH];

    int i = 0;
    while (i < length) {
        // Skip whitespace characters
        while (i < length && isspace(input[i])) {
            i++;
        }

        // Check for operators
        if (input[i] == '+' || input[i] == '--' || input[i] == '*' || input[i] == '/' ||
        input[i] == '=' || input[i] == '-' || input[i] == '%' || input[i] == '>') {
            printf("Operator: %c\n", input[i]);
            i++;
        }
        // Check for punctuation
        else if (input[i] == '(' || input[i] == ')' || input[i] == '{' || input[i] == '}' ||
        input[i] == ',' || input[i] == ';') {
            printf("Punctuation: %c\n", input[i]);
            i++;
        }
        // Check for constants
        else if (isdigit(input[i])) {
```

```c
            int j = 0;
            while (i < length && isdigit(input[i])) {
                token[j++] = input[i++];
            }
            token[j] = '\0';
            printf("Constant: %s\n", token);
        }
```

```
        // Check for string constants
        else if (input[i] == '"') {
            int j = 0;
            token[j++] = input[i++];  // Store the opening double quote

            while (i < length && input[i] != '"' && j < MAX_TOKEN_LENGTH - 1) {
                token[j++] = input[i++];
            }
            if (i < length && input[i] == '"') {
                token[j++] = input[i++];  // Store the closing double quote
                token[j] = '\0';
                printf("String Constant: %s\n", token);
            } else {
                printf("Invalid String Constant\n");
            }
        }
```

```
        // Check for identifiers or keywords
        else if (isalpha(input[i]) || input[i] == '_') {
            int j = 0;
            while (i < length && isValidIdentifierChar(input[i])) {
                token[j++] = input[i++];
            }
            token[j] = '\0';
            if (strcmp(token, "if") == 0 || strcmp(token, "else") == 0 ||
            strcmp(token, "while") == 0 || strcmp(token, "for") == 0 ||
            strcmp(token, "int") == 0) {
                printf("Keyword: %s\n", token);
            } else {
                printf("Identifier: %s\n", token);
            }
        }
```

```
        // Invalid token
        else {
            printf("Invalid token: %c\n", input[i]);
            i++;
        }
    }
}
int main() {
    const char* input = "int main() { int x = 5; while (x > 0) { printf(\"x = %d\\n\", x); x--; } return 0; }";
    tokenizeString(input);

    return 0;
}
```

OUTPUT:

```
Keyword: int
Identifier: main
Punctuation: (
Punctuation: )
Punctuation: {
Keyword: int
Identifier: x
Operator: =
Constant: 5
Punctuation: ;
Keyword: while
Punctuation: (
Identifier: x
Operator: >
Constant: 0
Punctuation: )
Punctuation: {
Identifier: printf
Punctuation: (
String Constant: "x = %d\n"
Punctuation: ,
Identifier: x
Punctuation: )
Punctuation: ;
Identifier: x
Operator: -
Operator: -
Punctuation: ;
Punctuation: }
Identifier: return
Constant: 0
Punctuation: ;
Punctuation: }


...Program finished with exit code 0
Press ENTER to exit console.
```

4. Write a C program for LL (1) Parsing.

1. E -> T E'
2. E' -> + T E'
3. E' -> epsilon
4. T -> F T' 5.
T' -> * F T' 6.
T' -> epsilon 7. F -> (E)
8. F -> id

CODE:

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
// Global variables
char input[100];
int position;
/* Function to check if the current symbol
matches the expected symbol*/
bool match(char expected) {
    if (input[position] == expected) {
        position++;
        return true;
    } else {
        return false;
    }
}
// Function to perform the parsing
bool E();
bool EPrime();
bool T();
bool TPrime();
bool F();

bool E() {
    if (T() && EPrime()) {
        return true;
    }
    return false;
}
```

```
bool EPrime() {
    int savedPosition = position;

    if (match('+') && T() && EPrime()) {
        return true;
    }

    position = savedPosition; // backtrack

    // Epsilon production
    return true;
}

bool T() {
    if (F() && TPrime()) {
        return true;
    }
    return false;
}
```

```
bool TPrime() {
    int savedPosition = position;

    if (match('*') && F() && TPrime()) {
        return true;
    }

    position = savedPosition; // backtrack

    // Epsilon production
    return true;
}
bool F() {
    if (match('(') && E() && match(')')) {
        return true;
    } else if (match('i')) {
        return true;
    }
    return false;
}

int main() {
    // Read the input from the user
    printf("Enter the input string: ");
    scanf("%s", input);

    // Perform the parsing
    if (E() && input[position] != '\0') {
        printf("The input string is valid.\n");
    } else {
        printf("The input string not is valid.\n");
    }
    return 0;
}
```

OUTPUT:

```
Enter the input string: id+id*id
The input string is valid.

--------------------------------
Process exited after 30.86 seconds with return value 0
Press any key to continue . . .
```