Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No. : CSE 4130

Course Name : Formal Languages and Compilers Lab

Assignment No. : 02

Submitted By:

Name :
ID No. : 17 01 04 012
Session : Spring - 2020
Section : A (A1)

QUESTION :

Suppose, we have a C source program scanned and filtered as it was done in Session 1. We now take that modified file as input, and separate the lexemes first. We further recognize and mark the lexemes as different types of tokens like keywords, identifiers, operators, separators, parenthesis, numbers, etc.

ANSWER:

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



bool validIdentifier(char   *str)
{

   int len=0,flg;

   if(strlen(str)>=1 && (str[0]>='a'&&str[0]<='z')||(str[0]>='A'&&str[0]<='Z')||(str[0]=='_') )
   {

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

      if(len==strlen(str))
```

```
        {
                flg=1;

        }

    }
    else if((strlen(str)==1) &&
((str[0]>='a'&&str[0]<='z')||(str[0]>='A'&&str[0]<='Z')||(str[0]=='_') ))
    {
        flg=1;

    }
Suppose, we have a C source program scanned and filtered as it was done in
Session 1. We now take that modified file as input, and separate the lexemes first. We
further
recognize and mark the lexemes as different types of tokens like keywords, identifiers,
operators, separators, parenthesis, numbers, etc.
    else
    {
        flg=0;
    }

    if(flg==1)
        return true;

    else
    {
        return false;

    }



}

// Returns 'true' if the string is a KEYWORD.
```

```
bool isKeyword(char   *str)
{
    if (!strcmp(str, "if") || !strcmp(str, "else") ||
            !strcmp(str, "while") || !strcmp(str, "do") ||
            !strcmp(str, "break") || !strcmp(str, "include")||
            !strcmp(str, "continue") || !strcmp(str, "int")
            || !strcmp(str, "double") || !strcmp(str, "float")
            || !strcmp(str, "return") || !strcmp(str, "char")
            || !strcmp(str, "case") || !strcmp(str, "char")
            || !strcmp(str, "sizeof") || !strcmp(str, "long")
            || !strcmp(str, "short") || !strcmp(str, "typedef")
            || !strcmp(str, "switch") || !strcmp(str, "unsigned")
            || !strcmp(str, "void") || !strcmp(str, "static")
            || !strcmp(str, "struct") || !strcmp(str, "goto") )

        return (true);
    return (false);
}

// Returns 'true' if the string is an INTEGER.
bool isInteger(char   *str)
{
    int i, len = strlen(str);
    for (i = 0; i < len; i++)
    {
        if (str[i] == '0' || str[i] == '1' && str[i] == '2'
                && str[i] == '3' && str[i] == '4' && str[i] =='5'
                && str[i] == '6' && str[i] == '7' && str[i] == '8'
                && str[i] == '9' )
            return (true);
    }
    return (false);
}
```

```
void check(char   *str)
{

    char ch ;

    printf("\n%s\n",str);
    char *temp = (char*)malloc(1000);
    bool flag = 0;

    int i,j, len = strlen(str);

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




        if (str[i] == ' ' || str[i] == '+' || str[i] == '-' || str[i] == '*' ||
                str[i] == '/' || str[i] == ',' || str[i] == ';' || str[i] == '>' ||
                ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
                str[i] == '[' || str[i] == ']' || str[i] == '{' || str[i] == '}'||
                str[i] == '\''||str[i]== '&'||str[i] == '|'||str[i] == '!'||str[i] == '#')
        {

            if(isKeyword(temp))
            {
                printf("'%s' ",temp);

            }
            else if(validIdentifier(temp))
            {

                printf("'%s' ",temp);


            }
```

```
            else if(isInteger(temp))
            {
                printf("'%s' ",temp);


            }

            if(str[i] != ' ')
            printf("'%c' ",str[i]);



            *temp = NULL;


        }

        else if (str[i] == '+' || str[i] == '-' || str[i] == '*' ||
                str[i] == '/' || str[i] == '>' || str[i] == '<' ||
                str[i] == '='||str[i] == '&'||str[i] == '|'||str[i] == '!')
        {

            if(isKeyword(temp))
            {
                printf("'%s' ",temp);


            }
            else if(validIdentifier(temp))
            {
                printf("'%s' ",temp);

            }
            else if(isInteger(temp))
            {
                printf("'%s' ",temp);
```

```
        }
        *temp = NULL;
        printf("'%c' ",str[i]);

    }
    else  if (str[i]== '\''||str[i] == ','||str[i] == '.')
    {

        if(isKeyword(temp))
        {
            printf("'%s' ",temp);


        }
        else if(validIdentifier(temp))
        {
            printf("'%s' ",temp);


        }
        else if(isInteger(temp))
        {
            printf("'%s' ",temp);


        }

        printf("'%c' ",str[i]);

        *temp = NULL;

    }

    else
```

```
        {

            strncat(temp,&str[i],1);

            //printf("%s",temp);



        }

    }
    strncat(temp,&str[i],1);


    if(isKeyword(temp))
    {
        printf("'%s' ",temp);

    }
    else if(validIdentifier(temp))
    {
        printf("'%s' ",temp);

    }
    else if(isInteger(temp))
    {
        printf("'%s' ",temp);

    }
    printf("'%s' ",temp);
}
```

```
int main ()
{
   FILE *fp;

   char str[10000];
   char *ch;
   fp=fopen("input.c","r");

   FILE *fp1, *fp2;
   char c, c1, c2, c3;

   fp1 = fopen("input.c", "r");
   fp2 = fopen("output.txt", "w");
   if(!fp1)
   {
      printf(" \n file can not be opened \n ");
   }
   else
   {
      while((c = fgetc(fp1)) != EOF)
      {

         if(c =='\n')
         {
         }
         else if(c == '/')
         {
            c1 = fgetc(fp1);
            if(c1 == '*')
            {
               while((c2 = fgetc(fp1)) != '*');
               if((c3 = fgetc(fp1)) == '/')
               {

               }
               else
```

```
                {
                    fputc(c3, fp2);
                }
            }
            else if(c1 == '/')
            {
                while((c2 = fgetc(fp1)) != '\n');
            }
            else
            {
                fputc(c, fp2);
                fputc(c1, fp2);
            }
        }
        else
            fputc(c, fp2);
    }
    fclose(fp1);
    fclose(fp2);
}
//new code
fp = fopen("output.txt", "r");

fgets(str,10000,fp);

ch=strtok(str,";");
printf("%s\n",ch);
printf(" \n in loop \n");
while(ch!=NULL)
{
    check(ch);
    ch = strtok(NULL, ";");


}
return 0;
```

```
}
/*

*/
```