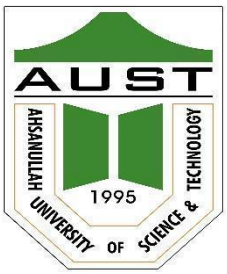

Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE4130**
Course Title : **Formal Languages and Compilers Lab**
Assignment No : **04**

Date of Performance :**27.12.2022**
Date of Submission : **14.02.2023**

Submitted To : **Mr. Aminur Rahman & Mr. Al Hasib Mahamud**

Submitted By-
Group : **A2**
Name : **Akila Maksud**
Id : **190104038**
Section : **A**

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

int isNumericConstant(char *lex)
{
   int i, l,s;
   i =0;
   if(isdigit(lex[i]))
   {
      s = 1;
      i++;
   }
   else if(lex[i]=='.')
   {
      s = 2;
      i++;
   }
   else
   {
      s =0;
   }
   l = strlen(lex);
   if(s==1)
   {
      for(; i<l; i++)
      {
         if(isdigit(lex[i]))
         {
            s = 1;
         }
         else if(lex[i]=='.')
         {
            s = 2;
            i++;
            break;
         }
         else
         {
            s = 0;
            break;
         }
      }
   }
   if(s==2)
   {
      if(isdigit(lex[i]))
      {
         s = 3;
         i++;
      }
      else
      {
         s = 0;
      }
   }
   if(s==3)
   {
      for(; i<l; i++)
      {
         if(isdigit(lex[i]))
         {
            s = 3;
         }
         else
         {
            s = 0;
            break;
         }
      }
   }
   if(s == 3)
   {
      s = 1;
      return 1;
   }
   return s;
}
int isSeparator(char *arr)
```

```
{
   int c = 0;
   if(strlen(arr) == 1 && arr[0] == ',' || arr[0] == ';')
   {
      c = 1;
   }
   return c;
}
int isParenthesis( char *arr)
{
   int c = 0;
   if(strlen(arr) == 1 && arr[0] == '(' || arr[0] ==')' || arr[0] =='{' || arr[0] == '}' || arr[0] == '\'' || arr[0] == '\"')
   {
      c = 1;
   }
   return c;
}
int isOperator( char *arr)
{
   int c = 0;
   if( strlen(arr) == 1 && arr[0] == '<' || arr[0]=='>' || arr[0]=='=' || arr[0] == '/' || arr[0] =='%' || arr[0] == '+' || arr[0] == '-' || arr[0] == '*')
   {
      c = 1;
   }
   else if( strlen(arr) == 2 && arr[0] == '<' || arr[0]=='>' || arr[0]=='+' || arr[0]=='/' || arr[1] == '-' || arr[1] == '*' )
   {
      if(arr[1]== '='  )
      {
         c = 1;
      }
   }
   return c;
}
int isKeyword(char *arr)
{
   int c = 0;
   char Keyword[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"
                        };
   for(int i =0; i<32; i++)
   {
      if(strcmp(Keyword[i],arr)==0)
      {
         c =1;
      }
   }
   return c;
}
int isIdentifier(char *arr)
{
   int c = 0, i =0, l;
   if((isalpha(arr[i])) || (arr[i] =='_'))
   {
      c = 1;
      i++;
   }

   else
   {
      c = 0;
   }
   l=strlen(arr);
   if(c == 1)
   {
```

```
      for(; i<l; i++)
      {
        if(isalpha(arr[i])  ||  arr[i]=='_'  ||
isdigit(arr[i]))
        {
          c = 2;
        }
        else
        {
          c = 0;
          break;
        }
      }
   }
   if( c == 2)
   {
     c = 1;
   }
   return c;
}
char
a[10000],b[10000],temp[10000],arr1[10
00][50];
int curr = 0;
struct table
{
   char *name;
   char *id_type;
   char *data_type;
   char *scope;
   char *value;
} tab[100];//for symbol table

void  insert(int  si,  char  *name,  char
*id_type, char *data_type, char *scope,
char *value)
{
   tab[si].name = name;
   tab[si].id_type=id_type;
   tab[si].data_type=data_type;
   tab[si].scope= scope;
   tab[si].value = value;
}
void display()
{
   int i =0;
   printf("SI.No\t\tName\t\tId
Type\t\tData Type\tScope\t\tValue\n");
   for(int i = 0; i<curr; i++)
   {
printf("%d\t\t%s\t\t%s\t\t%s\t\t%s\t\
t%s\n",i+1,tab[i].name,tab[i].id_type,tab[i
].data_type,tab[i].scope,tab[i].value);
   }
}
int  lookup(char  *name,char  *id_type,char
*scope)
{
   int i=0;
   for(i=0; i<curr; i++)
   {
     if(strcmp(tab[i].name,name)==0    &&
strcmp(tab[i].id_type,id_type)==0        &&
strcmp(tab[i].scope,scope)==0 )
     {
       return i;
     }
   }
   return -1;
}
void set_attribute(int curr, char *value)
{
   tab[curr].value = value;
}

char* check(char *arr)
{
   if(strlen(arr) == 1)
   {
     return arr;
```

```
    }
    else if(arr[0]>='0' && arr[0]<='9')
    {
        return arr;
    }
    else
    {
        return "";
    }
}

struct indx
{
    int id;
    char *name;
} indx[30];

void insert_index(int s, int id, char *name)
{
    indx[s].id = id;
    indx[s].name=name;
}
int id_number(char *arr)
{
    for(int i =0; i<curr; i++)
    {
        if(strcmp(arr,indx[i].name)==0)
        {
            return indx[i].id;
        }
    }
    return -1;
}
void free_space()
{
    for(int i =0; i<curr; i++)
    {
        indx[i].id='\0';
        indx[i].name='\0';
    }
    for(int j =0; j<curr; j++)
    {
        tab[j].data_type='\0';
        tab[j].id_type='\0';
        tab[j].name='\0';
        tab[j].scope='\0';
        tab[j].value='\0';
    }
}

char cfg_str[30];
int cfg = 0,c_len,ci =0;
void loop_stat()
{
    if(cfg_str[ci] == 'w' && cfg_str[ci+1] == 'h'&& cfg_str[ci+2] == 'i'&& cfg_str[ci+3] == 'l'&& cfg_str[ci+4] == 'e')
    {
        ci = ci+5;
        if(cfg_str[ci] == '(')
        {
            ci++;
            expn();
            if(cfg_str[ci] == ')')
            {
                ci++;
                stat();
                if(ci==c_len)
                    return;
                else
                {
                    cfg = 0;
                    return;
                }
            }
            else
                return;
        }
        else
        {
```

```
        cfg = 0;
        return;
      }
    }
    else if(cfg_str[ci] == 'f' && cfg_str[ci+1]
== 'o' && cfg_str[ci+2] == 'r')
    {
      ci = ci+3;
      if(cfg_str[ci] == '(')
      {
        ci++;
        asgn_stat();
        if(cfg_str[ci] == ';')
        {
          ci++;
          expn();
          if(cfg_str[ci] == ';')
          {
            ci++;
            asgn_stat();
            if(cfg_str[ci] == ')')
            {
              ci++;
              stat();
              if(ci==c_len)
                return;
              else
              {
                cfg = 0;
                return;
              }
            }
            else
            {
              cfg = 0;
              return;
            }

          }
          else
            {
              cfg = 0;
              return;
            }
          }
          else
          {
            cfg = 0;
            return;
          }
        }
        else
        {
          cfg = 0;
          return;
        }
      }
      else
      {
        cfg = 0;
        return;
      }
    }
    else
    {
      cfg = 0;
      return;
    }
}
void extn1()
{
    if((c_len-1) == ci)
    {
      cfg = 1;
      ci++;
      return;
    }
    else
    {
      if(cfg_str[ci] == 'e' && cfg_str[ci+1] ==
'l' && cfg_str[ci+2] =='s' || cfg_str[ci+3] ==
'e')
      {
        ci=ci+4;
        cfg=0;
        stat();
```

```
        if(cfg == 1)
          return;
        else
        {
          cfg = 0;
          return;
        }
      }
      else
      {
        cfg = 0;
        return;
      }
    }
}
void dscn_stat()
{
  if(cfg_str[ci] == 'i')
  {
    ci++;
    if(cfg_str[ci] == 'f')
    {
      ci++;
      if(cfg_str[ci] == '(')
      {
        ci++;
        expn();
        if(cfg_str[ci] == ')')
        {
          ci++;
          stat();
          if(ci==c_len)
            return;
          else
          {
            if(cfg == 1)
            {
              extn1();
            }
            else
              return;
          }
        }
      }
    }
  }
  else
  {
    cfg = 0;
    return;
  }
}
void F()
{
  if(isdigit(cfg_str[ci]))
  {
    ci++;
    cfg = 1;
    return;
  }
  else if(cfg_str[ci] == 'a' || cfg_str[ci] == 'b'
|| cfg_str[ci] == 'c' || cfg_str[ci] == 'd' ||
cfg_str[ci] == 'e')
  {
    ci++;
    cfg = 1;
    return;
  }
  else if(cfg_str[ci] == '(')
  {
    ci++;
    E();
    ci++;
    if(cfg_str[ci] == ')')
    {
      cfg = 1;
      return;
    }
  }
}
```

```
void T()
{
   F();
   if(ci==c_len)
      return;
   if(ci<c_len-1)
   {
      if(cfg_str[ci] == '*' || cfg_str[ci] == '/')
      {
         ci++;
         F();
      }
      else if(cfg == 1)
      {
         return;
      }
   }
}
void E()
{
   T();

   if(ci == c_len)
      return;
   if(ci < c_len-1)
   {
      if(cfg_str[ci] == '+' || cfg_str[ci] == '-')
      {
         ci++;
         T();
      }
      else if(cfg == 1)
      {
         return;
      }
   }

}
void smpl_expn()
{
   E();
   if(cfg == 1 && c_len==ci)
   {
      return;
   }
   else
      return;
}
void relop()
{
   if(cfg_str[ci] == '=')
   {
      ci++;
      if(cfg_str[ci] == '=')
      {
         cfg  = 1;
         return;
      }
      else
      {
         cfg = 0;
         return;
      }
   }
   else if(cfg_str[ci] == '!')
   {
      ci++;
      if(cfg_str[ci] == '=')
      {
         cfg  = 1;
         return;
      }
      else
      {
         cfg = 0;
         return;
      }
   }
   else if(cfg_str[ci] == '<')
   {
```

```
   ci++;
   cfg = 1;
   if(cfg_str[ci] == '=')
   {
      cfg  = 1;
      return;
   }
   else
   {
      return;
   }
}
else if(cfg_str[ci] == '>')
{
   ci++;
   cfg = 1;
   if(cfg_str[ci] == '=')
   {
      ci++;
      cfg  = 1;
      return;
   }
   else
   {
      return;
   }
}
else if(cfg_str[ci] == '>')
{
   ci++;
   cfg  = 1;
   return;
}
else if(cfg_str[ci] == '<')
{
   ci++;
   cfg  = 1;
   return;
}
else
   {
      cfg = 0;
      return;
   }
}
void extn()
{
   if((c_len-1) ==  ci)
   {
      cfg = 1;
      ci++;
      return;
   }
   else
   {
      relop();
      if(cfg == 1)
      {
         smpl_expn();
         if(c_len == ci)
            return;
      }
      else
         return;
   }
}
void expn()
{
   smpl_expn();
   if(c_len == ci)
   {
      return;
   }
   else
   {
      if(cfg == 1)
      {
         extn();
         return;
      }
```

```
   }
}
void asgn_stat()
{
   if(cfg_str[ci] == 'a' || cfg_str[ci] == 'b' ||
cfg_str[ci] == 'c'|| cfg_str[ci] == 'd'||
cfg_str[ci] == 'e')
   {
      ci++;
      if(cfg_str[ci] == '=')
      {
         ci++;
         expn();
         if(cfg ==1 && ci==c_len)
         {
            return;
         }
         else
         {
            cfg=1;
            return;
         }
      }
   }
   else
   {
      cfg = 0;
      return;
   }
}
void stat()
{
   int as = 0;
   asgn_stat();
   as = 1;
   if(cfg ==1 && (c_len==ci))
   {
      return;
   }
   else if(cfg==1)
      return;
   if(as == 1 && cfg == 0)
   {
      dscn_stat();
      if(cfg == 0)
      {
         loop_stat();
      }
   }
}
void searchCFG()
{
   FILE *p1,*p2;
   char line[500],buffer[50];
   int count=0;
   p1 = fopen("inter.txt", "r");
   p2 = fopen("cfg_found.txt", "w");
   while(fgets(line, sizeof(line), p1)) {
      if(strstr(line,"for")                    ||
strstr(line,"while"))
      {
         int si=0,startWriting=0;
         for(si=0;si<strlen(line);si++)
          {
            if(isdigit(line[si])                 &&
startWriting==0)
               fputc(line[si],p2);
            if(line[si]=='w'|| line[si]=='f')
            {
               startWriting=1;
               fputc(' ',p2);
            }
            if(startWriting && line[si]!=' ' &&
line[si]!=';')
            {
              fputc(line[si],p2);
            }
          }
      }
count++;
```

```
}
fclose(p1);
fclose(p2);
}
int main(void)
{
   FILE *p1, *p2, *p3, *p4, *p5, *p6;
   char c;
   int flag = 1, i =0, j = 0, k = 0, l = 0, t = 0, a,
b=0;
   char array[15];
   array[0]=' ';
   p1 = fopen("input.c","r");
   p2 = fopen("output.txt","w");
   if(!p1)
   {
      printf("\nFile Cannot be opened!");
   }
   else
   {
      while((c=fgetc(p1)) != EOF)
      {
         if( c == '/' && i ==0)
         {
            if( j == 1)
            {
               j = 0;
            }
            flag = 0;
            i = 1;
         }
         else if(i==1 && c=='/')
         {
            if( j == 1)
            {
               j = 0;
            }
            flag = 0;
            i =2;
         }
         else if( i ==2 && c=='\n')
         {
            if( j == 1)
            {
               j = 0;
            }

            flag = 1;
            i = 0;
            continue;
         }
         else if( i==1 && c=='*')
         {
            flag = 0;
            i =3;
         }
         else if( i ==3 && c =='/')
         {
            if( j == 1)
            {
               j = 0;
            }
            flag = 1;
            i = 0;
            continue;
         }
         else if(c == '\n')
         {
            if( j == 1)
            {
               j = 0;
            }
            continue;
         }
         else if ( c == '\t')
         {
            fputc(' ', p2);
            if( j == 1)
            {
               j = 0;
```

```
        }
        continue;
      }
      else if ( c == ' ')
      {

        if( j == 1)
        {
          j == 2;
          continue;
        }
        else if(j == 2)
        {
          j = 0;
          continue;
        }
        else
        {
          j = 1;
        }
      }
      else
      {
        if( j == 1)
        {
          j = 0;
        }
      }
      if(flag == 1)
      {
        fputc(c,p2);
      }
    }
}
fclose(p1);
fclose(p2);
p1 = fopen("input.c","r");
printf("Input C File :\n");
while((c = fgetc(p1)) != EOF)
{
    printf("%c",c);
}
printf("\n\n\n");
p2 = fopen("output.txt","r");
printf("Assignment 1's outtput:\n\n");
while((c = fgetc(p2)) != EOF)
{
    printf("%c",c);
}
printf("\n\n\n");
fclose(p2);
p2 = fopen("output.txt","r");
p3 = fopen("separated.txt", "w");

if(!p2)
{
    printf("\nFile Cannot be opened!");
}
else
{
    int p = 0, s=0;
    while((c=fgetc(p2)) != EOF)
    {
        if ( c == '(' || c ==')' || c =='{' || c == '}'
|| c == '\'' || c == '\"' || c ==',' || c ==';' || c
=='[' || c==']')
        {
            if(k == 1)
            {
                k = 0;
            }
            if( p == 0)
            {
                fputc(' ',p3);
                fputc(c,p3);
                fputc(' ',p3);
                p = 1;
            }
            else
            {
```

```
            fputc(c,p3);
            fputc(' ',p3);
            p = 1;
          }
        }
        else if( c == ' ')
        {
          if(p == 1)
          {
            continue;
          }
          else
          {
            fputc(c,p3);
            s = 1;
          }
          p = 0;
        }
        else if( c == '<' || c =='>' )
        {
          fputc(' ', p3);
          fputc(c,p3);
          k = 1;
        }
        else if ( c == '=' || c =='+' || c =='-' || c
=='*' || c=='/' || c=='%')
        {
          if(k == 1)
          {
            fputc(c,p3);
            fputc(' ',p3);
            k = 0;
            p = 1;
            s = 0;
          }
          if(s == 1)
          {
            fputc(c,p3);
            fputc(' ',p3);
            s = 0;
            p = 1;
          }
          else
          {
            fputc(' ', p3);
            fputc(c,p3);
            fputc(' ',p3);
            p = 1;
          }
        }
        else
        {
          if(k == 1)
          {
            fputc(' ',p3);
            k = 0;
          }
          if( p ==1)
          {
            p=0;
          }
          if (s== 1)
          {
            s = 0;
          }
          fputc(c,p3);
        }
      }

      fclose(p2);
      fclose(p3);
      p3 = fopen("separated.txt", "r");
      printf("Separated Lexems are
here:\n\n");
      while((c = fgetc(p3)) != EOF)
      {
        printf("%c",c);
      }
      printf("\n\n\n");
      p3 = fopen("separated.txt", "r");
```

```
    p4 = fopen("result.txt","w");
    if(!p3)
    {
       printf("\nFile Cannot be opened!");
    }
    else
    {
       while( (c = fgetc(p3)) != EOF )
       {
          if(c == ' ')
          {
             array[i] = '\0';
             i=0;
             if(isSeparator(array))
             {
                fputs("[Separator ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
             else if( isParenthesis(array))
             {
                fputs("[Parenthesis ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
             else if( isOperator(array))
             {
                fputs("[Operator ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
             else if( isNumericConstant(array))
             {
                fputs("[Numeric Constant ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
             else if( isKeyword(array))
             {
                fputs("[Keyword ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
             else if( isIdentifier(array))
             {
                fputs("[Identifier ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
             else
             {
                fputs("[Unknown ", p4);
                fputs(array, p4);
                fputs("] ", p4);
             }
          }
          else
          {
             array[i] = c;
             i++;
          }
       }
       arr1[a][b]='\0';
    }
  }
  fclose(p3);
  fclose(p4);
  printf("\n\n");
  printf("Categorized Lexemes are here: \n\n");
  p4 = fopen("result.txt", "r");
  while((c = fgetc(p4)) != EOF)
  {
     printf("%c",c);
  }
  printf("\n\n\n");
  fclose(p4);
  p3 = fopen("separated.txt", "r");
```

```
p5 = fopen("identifier.txt","w");
a = 0;
if(!p3)
{
   printf("\nFile Cannot be opened!");
}
else
{
   while( (c = fgetc(p3)) != EOF )
   {
      if(c == ' ')
      {
         array[i] = '\0';
         i=0;
         if( isKeyword(array))
         {
            arr1[a][b]='\0';
            a++;
            b =  0;
            fputs("[", p5);
            fputs(array, p5);
            fputs("] ", p5);

         }
         else if( isIdentifier(array))
         {
            fputs("[Identifier ", p5);
            fputs(array, p5);
            fputs("] ", p5);
            arr1[a][b]='\0';
            a++;
            arr1[a][0]='i';
            arr1[a][1]='d';
            arr1[a][2]='\0';
            a++;
            b = 0;
         }
         else
         {
            arr1[a][b]='\0';
            a++;
            b =  0;
            fputs("[", p5);
            fputs(array, p5);
            fputs("] ", p5);
         }
      }
      else
      {
         array[i] = c;
         arr1[a][b]= c;
         b++;
         i++;
      }
   }
   fclose(p3);
   fclose(p5);
   printf("Categorized Indentifiers are here: \n\n");
   p5 = fopen("identifier.txt", "r");
   while((c = fgetc(p5)) != EOF)
   {
      printf("%c",c);
   }
   printf("\n\n\n");
   fclose(p5);
   FILE *fp3;
   p3 = fopen("separated.txt", "r");
   p6 = fopen("output.txt","w");
   if(!p3)
   {
      printf("\nFile Cannot be opened!");
   }
   else
   {
      while( (c = fgetc(p3)) != EOF )
      {
         if(c == ' ')
         {
```

```
            array[i] = '\0';
            i=0;
            if(isKeyword(array))
            {
               fputs("[", p6);
               fputs(array, p6);
               fputs("] ", p6);
            }
            else if(isIdentifier(array))
            {
               int id = id_number(array);
               fputs("[Identifier ", p6);
               fputc(id+1+48, p6);
               fputs("] ", p6);
               fputs(" ",p6);
            }
            else
            {
               fputs("[", p6);
               fputs(array, p6);
               fputs("] ", p6);
            }
         }
         else
         {
            array[i] = c;
            i++;
         }
      }
   }
   fclose(p3);
   fclose(p6);
   printf("\n\n\nModified token streams
are here: \n\n");
   p6 = fopen("output.txt", "r");
   while((c = fgetc(p6)) != EOF)
   {
      printf("%c",c);
   }
   printf("\n\n\n");
   fclose(p6);
   FILE *fp,*fp1, *fp2 ;
   char narray[1000][1000];
   int flag1 =0;
   p1 = fopen("input.c","r");
   fp=fopen("inter.txt","w");
   if(!p1)
      printf("\nFile can't be opened!");
   int c1=0, c2=0;
   int line=1, sp =0;
   i =0;
   while(fgets(narray[i],500,p1))
   {
      fprintf(fp,"%d ",line);
      line++;
      for(int j=0; j<strlen(narray[i]); j++)
      {
         if(c2 ==0 && c1 ==0 &&
narray[i][j]=='/')
         {
            flag1 = 1;
            c1 = 1;
         }
         else if( c1 == 1 && narray[i][j]=='*')
         {
            flag1 =1;
            c2 =1;
         }
         else if( c2 ==1 && narray[i][j] =='/')
         {
            flag1 =0;
            c2 =0;
            c1=0;
         }
         else if(narray[i][j]== '\t')
         {
            if(sp ==0)
            {
               fputc(' ',fp);
               sp=1;
```

```
            }
            else if(sp ==1)
            {
               continue;
            }
         }
         else if(narray[i][j]== ' ')
         {
            if(sp ==0)
            {
               fputc(' ',fp);
               sp=1;
            }
            else if(sp ==1)
            {
               continue;
            }
         }
         else
         {
            if( sp ==1)
            {
               sp =0;
            }
            if(flag1 ==0 )
            {
               fputc(narray[i][j],fp);
            }
         }
      }
   }
   i++;
   c1 = 0;
   fclose(p1);
   fclose(fp);
   printf("\n\nInput c file with line
numbers: \n\n");
   fp = fopen("inter.txt", "r");
   while((c = fgetc(fp)) != EOF)
   {
      printf("%c",c);
   }
   printf("\n\n\n");
   fclose(fp);
   fp = fopen("inter.txt","r");
   fp1 = fopen("intermediate.txt", "w");
   if(!fp)
   {
      printf("\nFile Cannot be opened!");
   }
   else
   {
      int p = 0, s=0;
      i =0;
      while(fgets(narray[i],500,fp))
      {
         for(int j =0 ; j<strlen(narray[i]); j++)
         {
            if ( narray[i][j] == '(' || narray[i][j]
==')' || narray[i][j] =='{' || narray[i][j] == '}'
|| narray[i][j] == '\'' || narray[i][j] == '\"' ||
narray[i][j] ==',' || narray[i][j] ==';' ||
narray[i][j] =='[' || narray[i][j] ==']')
            {
               if(k == 1)
               {
                  k = 0;
               }
               if( p == 0)
               {
                  fputc(' ',fp1);
                  fputc(narray[i][j],fp1);
                  fputc(' ',fp1);
                  p = 1;
               }
               else
               {
                  fputc(narray[i][j],fp1);
                  fputc(' ',fp1);
                  p = 1;
```

```
            }
        }
        else if( narray[i][j] == ' ')
        {
            if(p == 1)
            {
                continue;
            }
            else if( j==1 || j==2)
            {
                if ( narray[i][j+1] == '(' || narray[i][j+1] ==')' || narray[i][j+1] =='{' || narray[i][j+1] == '}' || narray[i][j+1] == '\'' || narray[i][j+1] == '\"' || narray[i][j+1] ==',' || narray[i][j+1] ==';' || narray[i][j+1] =='[' || narray[i][j+1] ==']')
                {
                    continue;
                }
                else
                {
                    fputc(narray[i][j],fp1);
                    s = 1;
                }
            }
            else
            {
                fputc(narray[i][j],fp1);
                s = 1;
            }
            p = 0;
        }
        else if( narray[i][j] == '<' || narray[i][j] =='>' )
        {
            fputc(' ', fp1);
            fputc(narray[i][j],fp1);
            k = 1;
        }
        else if ( narray[i][j] == '=' || narray[i][j] =='+' || narray[i][j] =='-' || narray[i][j] =='*' || narray[i][j]=='/' || narray[i][j]=='%')
        {
            if(k == 1)
            {
                fputc(narray[i][j],fp1);
                fputc(' ',fp1);
                k = 0;
                p = 1;
                s = 0;
            }
            if(s == 1)
            {
                fputc(narray[i][j],fp1);
                fputc(' ',fp1);
                s = 0;
                p = 1;
            }
            else
            {
                fputc(' ', fp1);
                fputc(narray[i][j],fp1);
                fputc(' ',fp1);
                p = 1;
            }
        }
        else
        {
            if(k == 1)
            {
                fputc(' ',fp1);
                k = 0;
            }
            if( p ==1)
            {
                p=0;
            }
            if (s== 1)
```

```
            {
               s = 0;
            }
            fputc(narray[i][j],fp1);
          }
        }
      }
      fclose(fp);
      fclose(fp1);
      fp1 = fopen("intermediate.txt", "r");
      printf("New Separated Lexems are here:\n\n");
      while((c = fgetc(fp1)) != EOF)
      {
         printf("%c",c);
      }
      printf("\n\n\n");
   }
   fclose(fp1);
   fp1 = fopen("intermediate.txt", "r");
   fp2 = fopen("new_catagorized.txt","w");
   fp3 = fopen("errors.txt","w");
   char array3[15];
   array3[0]=' ';
   int n=0, r=0,spratr =0, forl=0, br1 =0, br2=0, key=0, elif =0, inden=0, rtrn =0;
   char array4[40][100][15];
   int pt=0;
   int lizesize[500];

   if(!fp1)
   {
      printf("\nFile Cannot be opened!");
   }
   else
   {
      while(fgets(narray[n],500,fp1) )
      {
         for(int j =0; j<strlen(narray[n]); j++)
         {
            if(narray[n][j] == ' ')
            {
               array3[r] = '\0';
               r=0;
               if(j == 1|| j==2)
               {
                  fputs(array3, fp2);
                  fputs(" ", fp2);
strcpy(array4[n][pt++],array3);
               }
               else if(isSeparator(array3))
               {
                  fputs("Separator ", fp2);
                  fputs(array3, fp2);
                  fputs(" ", fp2);
                  if(key == 1)
                  {
                     key = 0;
                  }
                  if(rtrn ==1)
                  {
                     rtrn =0;
                  }
                  if(spratr ==0)
                  {
                     spratr =1;
                  }
                  else if(spratr==1)
                  {
                     if(forl ==1)
                     {
                        spratr =1;
                        forl=0;
                     }
                     else
                     {
                        fputs("Duplicate token at line ",fp3);
                        fprintf(fp3,"%d",n+1);
                        fputc('\n',fp3);
```

```
                }
            }
strcpy(array4[n][pt++],array3);
        }
        else if( isParenthesis(array3))
        {
            fputs("Parenthesis ", fp2);
            fputs(array3, fp2);
            fputs(" ", fp2);
            if(spratr ==1)
            {
                spratr =0;
            }
            if(rtrn ==1)
            {
                rtrn =0;
            }
            if(strcmp(array3,"{")==0)
            {
                br2++;
            }
            if(key == 1)
            {
                key = 0;
            }
            if(strcmp(array3,"}")==0)
            {
                if(br2>0)
                {
                    br2--;
                }
                else if (br2 ==0)
                {
                    fputs("Misplaced } at line
",fp3);
                    fprintf(fp3,"%d",n+1);
                    fputc('\n',fp3);
                }
            }
            if(strcmp(array3,"(")==0)
            {
                br1++;
            }
            if(strcmp(array3,")")==0)
            {
                if(br1>0)
                {
                    br1--;
                }
                else
                {
                    fputs("Misplaced ) at line
",fp3);
                    fprintf(fp3,"%d",n+1);
                    fputc('\n',fp3);
                }
            }
strcpy(array4[n][pt++],array3);
        }
        else if( isOperator(array3))
        {
            fputs("Operator ", fp2);
            fputs(array3, fp2);
            fputs(" ", fp2);
            if(spratr ==1)
            {
                spratr =0;
            }
            if(rtrn ==1)
            {
                rtrn =0;
            }
strcpy(array4[n][pt++],array3);
        }
        else if(
isNumericConstant(array3))
        {
            fputs("Numeric Constant ",
fp2);
            fputs(array3, fp2);
```

```
                fputs(" ", fp2);
                if(spratr ==1)
                {
                  spratr =0;
                }
                if(rtrn ==1)
                {
                  rtrn =0;
                }
strcpy(array4[n][pt++],array3);
              }
              else if( isKeyword(array3))
              {
                fputs("Keyword ", fp2);
                fputs(array3, fp2);
                fputs(" ", fp2);
                if(spratr ==1)
                {
                  spratr =0;
                }
                if(strcmp(array3,"for")==0)
                {
                  forl =1;
                }
                if(key ==0)
                {
                  key=1;
                }
                else if(key ==1)
                {
if(strcmp(array3,"return")==0)
                  {

                    if(rtrn ==1)
                    {
                      rtrn =0;
                      key =0;
                    }
                  }
                  else
                  {
                    fputs("Duplicate token at
line ",fp3);
                    fprintf(fp3,"%d",n+1);
                    fputc('\n',fp3);
                  }
                }
                if(strcmp(array3,"if")==0)
                {
                  elif++;
                  rtrn =1;
                }
                if(strcmp(array3,"else")==0)
                {
                  if(elif>0)
                  {
                    elif--;
                    rtrn = 1;
                  }
                  else
                  {
                    fputs("""else""     without
previous ""if"" at line ",fp3);
                    fprintf(fp3,"%d",n+1);
                    fputc('\n',fp3);
                  }
                }
strcpy(array4[n][pt++],array3);
              }
              else if( isIdentifier(array3))
              {
                fputs("Identifier ", fp2);
                fputs(array3, fp2);
                fputs(" ", fp2);
                if(spratr ==1)
                {
                  spratr =0;
                }
                if(rtrn ==1)
                {
```

```
            rtrn =0;
          }
          if(key == 1)
          {
            key=0;
          }
          strcpy(array4[n][pt++],"id");
        strcpy(array4[n][pt++],array3);
        }
      }
      else
      {
        array3[r] = narray[n][j];
        r++;
      }
    }
    lizesize[n]=pt;
    n++;
    pt=0;
}
if(br2 >0)
{
    while(br2 !=0)
    {
      fputs("Expected } at line ",fp3);
      fprintf(fp3,"%d",n+1);
      fputc('\n',fp3);
      br2--;
    }
}
if(br1 >0)
{
    while(br1 !=0)
    {
      fputs("Expected ) at line ",fp3);
      fprintf(fp3,"%d",n+1);
      fputc('\n',fp3);
      br1--;
    }
}
    }
    char * scope1 = "global";
    int s=0;
    for(int e=0; e<n; e++)
    {
      for(int f =0; f<lizesize[e]; f++)
      {
        if(strcmp(array4[e][f],"id")==0)
        {
          if(strcmp(array4[e][f+2],"(")==0)
          {
            if(!(strcmp(array4[e][f-
1],"int")==0      ||      strcmp(array4[e][f-
1],"double")==0   ||   strcmp(array4[e][f-
1],"float")==0     ||     strcmp(array4[e][f-
1],"char")==0     ||     strcmp(array4[e][f-
1],"void")==0))
              {
                int          sr          =
lookup(array4[e][f+1],"func","global");
                if(sr==-1)
                {
                  fprintf(fp3,"Expected
declaration of function %s at line
%d\n",array4[e][f+1],e);
                }
                f=f+3;
              }
              else
              {
insert(curr,array4[e][f+1],"func",array4[e
][f-1],"global","          ");
insert_index(s++,curr,array4[e][f+1]);
                curr++;
                scope1 = array4[e][f+1];
                f=f+3;
              }
            }
            else
if(strcmp(array4[e][f+2],"=")==0)
```

```
        {
          if(strcmp(array4[e][f-1],"int")==0 || strcmp(array4[e][f-1],"double")==0 || strcmp(array4[e][f-1],"float")==0 || strcmp(array4[e][f-1],"char")==0 || strcmp(array4[e][f-1],"void")==0)
          {
            int sr = lookup(array4[e][f+1],"var",scope1);
            if(sr!=-1)
            {
              fprintf(fp3,"ID %s at line %d already declared in %s scope\n",array4[e][f+1],e+1,scope1);
            }
            else
            {
insert(curr,array4[e][f+1],"var",array4[e][f-1],scope1,check(array4[e][f+3]));

insert_index(s++,curr,array4[e][f+1]);
              curr++;
              f=f+3;
            }
          }
          else
          {
            int sr = lookup(array4[e][f+1],"var",scope1);
            if(sr==-1)
            {
              fprintf(fp3,"Expected declaration of Identifier %s at line %d\n",array4[e][f+1],e+1);
            }
            else
            {

set_attribute(sr,check(array4[e][f+3]));
insert_index(s++,sr,array4[e][f+1]);
            }
          }
        }
        else if(strcmp(array4[e][f+2],";")==0 || strcmp(array4[e][f+2],",")==0 || strcmp(array4[e][f+2],")")==0)
        {
          if(strcmp(array4[e][f-1],"int")==0 || strcmp(array4[e][f-1],"float")==0 || strcmp(array4[e][f-1],"double")==0 || strcmp(array4[e][f-1],"char")==0 )
          {
            int val = lookup(array4[e][f+1],"var",scope1);
            if(val == -1)
            {
insert(curr,array4[e][f+1],"var",array4[e][f-1],scope1,"");
insert_index(s++,curr,array4[e][f+1]);
              curr++;
              f +=2;
            }
            else
            {
              fprintf(fp3,"ID %s at line %d already declared in %s scope\n",array4[e][f+1],e+1,scope1);
            }
          }
        }
        else if(strcmp(array4[e][f],"}")==0)
        {
          scope1 = "global";
        }
      }
```

```
  }
  fclose(fp1);
  fclose(fp2);
  fclose(fp3);
  printf("\n\nSymbol Table:\n\n");
  display();
  printf("\n\n");
  printf("New Categorized Lexemes are here: \n\n");
  fp2 = fopen("new_catagorized.txt", "r");
  while((c = fgetc(fp2)) != EOF)
  {
    printf("%c",c);
  }
  printf("\n\n\n");
  fclose(fp2);
  printf("\n\n\nDetected Errors are here: \n\n");
  fp3 = fopen("errors.txt", "r");
  while((c = fgetc(fp3)) != EOF)
  {
    printf("%c",c);
  }
  printf("\n\n\n");
  fclose(fp3);
  free_space();
  printf("CFG Parsing: \n");
  searchCFG();
  FILE *ptr,*cp1;
  cp1 = fopen("cfg_errors.txt","w");
  if ((ptr = fopen("inter.txt","r")) == NULL)
  {
    printf("Error! opening file");
  }
  int check_line,error_line;
  while(fscanf(ptr,"%s",&cfg_str)>0)
  {
    check_line=atoi(cfg_str);
    if(check_line> 0)
    {
      error_line=check_line;
      continue;
    }
    ci = 0;
    c_len = strlen(cfg_str);
    if(c_len>=1)
    {
      stat();
    }
    else
      fprintf(p1,"Please find another string %s.\n",cfg_str);
    if(c_len == ci && cfg == 1)
      fprintf(cp1,"At line %d: %s is valid.\n",error_line,cfg_str);
    else
      fprintf(cp1,"At line %d: %s is not valid.\n",error_line,cfg_str);
  }
  fclose(ptr);
  fclose(cp1);
  printf("\n");
  char ch;
  cp1 = fopen("cfg_errors.txt","r");
  do
  {
    ch = fgetc(p1);
    putchar(ch);
  }
  while(ch != EOF);

  fclose(cp1);
  return 0;
}
```