**PRACTICAL-4**

**Aim : Implementation of lexical analyser using lex tool.**

**Code:**

%{

#include<stdio.h>

int comment=0;

%}

ident [a-zA-Z][a-zA-Z]\*

%%

#.\* {printf("\nHeader files: %s",yytext); }

auto |

case |

char |

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 { printf("\nKeyword: %s",yytext);}

"/\*" {comment=1;}

"\*/" { comment=0;}

{ident}\( {if(!comment)printf("\nFUNCTION: %s",yytext);}

{ident}(\[[0-9]\*\])? {if(!comment)printf("\nIdentifier: %s",yytext); }

\".\*\" {if(!comment)printf("\nString: %s",yytext);}

[0-9]+ {if(!comment)printf("\nConstant: %s",yytext);}

= {if(!comment)printf("\nAssignment operator: %s",yytext);}

\<= |

\>= |

\< |

\> |

== { if(!comment) printf("\nRelational operator: %s",yytext);}

\+ |

\- |

\/ |

\\* |

\& |

% { if(!comment) printf("\nOperator: %s",yytext);}

\@ |

\$ |

\( |

\) |

\; |

\: |

\{ |

\} |

\, { if(!comment) printf("\nSpecial character: %s",yytext);}

%%

int main()

{

FILE \*f;

f=fopen("abc.txt","r");

yyin=f;

yylex();

return 0;

}

int yywrap()

{

return 0;

}

**Input : abc.txt**

void main {

int a, b, c;

a = 30;

b = 20;

c = a - b;

printf("%d", & c)

}

**Output**

