**PRACTICAL – 3**

Write a C program to test whether a given identifier is valid or not.

SOURCE CODE: -

#include<stdio.h>

#include<conio.h>

#include<ctype.h>

void main()

{

char a[10];

int flag, i=1;

clrscr();

printf("\n Enter an identifier:");

gets(a);

if(isalpha(a[0]))

flag=1;

else

printf("\n Not a valid identifier");

while(a[i]!='\0')

{

if(!isdigit(a[i])&&!isalpha(a[i]))

{

flag=0;

break;

}

i++;

}

if(flag==1)

printf("\n Valid identifier");

getch();

}

OUTPUT :-

Enter an identifier: first

Valid identifier

Enter an identifier:1aqw

Not a valid identifier

**PRACTICAL – 4**

Implementation of recursive descent parser.

SOURCE CODE :-

recursive.c

#include<stdio.h>

#include<string.h>

#include<ctype.h>

char input[10];

int i,error;

void E();

void T();

void Eprime();

void Tprime();

void F();

main()

{

i=0;

error=0;

printf("Enter an arithmetic expression : "); // Eg: a+a\*a

gets(input);

E();

if(strlen(input)==i&&error==0)

printf("\nAccepted..!!!\n");

else printf("\nRejected..!!!\n");

}

void E()

{

T();

Eprime();

}

void Eprime()

{

if(input[i]=='+')

{

i++;

T();

Eprime();

}

}

void T()

{

F();

Tprime();

}

void Tprime()

{

if(input[i]=='\*')

{

i++;

F();

Tprime();

}

}

void F()

{

if(isalnum(input[i]))i++;

else if(input[i]=='(')

{

i++;

E();

if(input[i]==')')

i++;

else error=1;

}

else error=1;

          }

OUTPUT :-

a+(a\*a) a+a\*a , (a), a , a+a+a\*a+a.... etc are accepted

++a, a\*\*a, +a, a, ((a . . . etc are rejected.

**PRACTICAL – 5**

Implementation of Lexical Analyzer using Lex.

SOURCE CODE :-

%{

int COMMENT=0;

%}

identifier [a-zA-Z][a-zA-Z0-9]\*

%%

#.\* {printf("\n%s is a preprocessor directive",yytext);}

int |

float |

char |

double |

while |

for |

struct |

typedef |

do |

if |

break |

continue |

void |

switch |

return |

else |

goto {printf("\n\t%s is a keyword",yytext);}

"/\*" {COMMENT=1;}{printf("\n\t %s is a COMMENT",yytext);}

{identifier}\( {if(!COMMENT)printf("\nFUNCTION \n\t%s",yytext);}

\{ {if(!COMMENT)printf("\n BLOCK BEGINS");}

\} {if(!COMMENT)printf("BLOCK ENDS ");}

{identifier}(\[[0-9]\*\])? {if(!COMMENT) printf("\n %s IDENTIFIER",yytext);}

\".\*\" {if(!COMMENT)printf("\n\t %s is a STRING",yytext);}

[0-9]+ {if(!COMMENT) printf("\n %s is a NUMBER ",yytext);}

\)(\:)? {if(!COMMENT)printf("\n\t");ECHO;printf("\n");}

\( ECHO;

= {if(!COMMENT)printf("\n\t %s is an ASSIGNMENT OPERATOR",yytext);}

\<= |

\>= |

\< |

== |

\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR",yytext);}

%%

int main(int argc, char \*\*argv)

{

FILE \*file;

file=fopen("var.c","r");

if(!file)

{

printf("could not open the file");

exit(0);

}

yyin=file;

yylex();

printf("\n");

return(0);

}

int yywrap()

{

return(1);

}

INPUT:

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b,c;

a=1;

b=2;

c=a+b;

printf("Sum:%d",c);

}

OUTPUT :-

