13.Program to generate anbn grammar using yacc.

Source code:

anbn.l

%{

#include<stdio.h>

#include"y.tab.h"

%}

%%

a return A;

b return B;

\n|. return yytext[0];

%%

anbn.y

%{

#include<stdio.h>

int vd;

%}

%union

{

char dval;

}

%token <dval> A

%token <dval> B

%%

str: s '\n' { vd=1; return 0;}

s : A s B ;

| ;

%%

int main()

{

printf("enter the string:");

yyparse();

if(vd==1)

printf(" \nvalid string\n");

else

printf("\n not a valid string\n");

}

yyerror(char \*s)

{

printf("%s",s);

}

Testing:

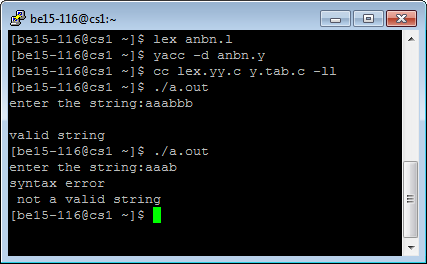
Input:

Enter the string:aabb

Output:

Valid string

Actual Output:



Result:

12.program to implement calculator using yacc.

Source code:

Calculator.l

%{

#include<stdio.h>

#include"y.tab.h"

%}

%%

[0-9]+ {yylval.dval=atoi(yytext); return DIGIT;}

\n|. return yytext[0];

%%

Calculator.y

%{

/\*\*/

%}

%union

{

int dval;

}

%token <dval> DIGIT

%type <dval> expr

%type <dval> expr1

%%

line : expr '\n' {printf("%d\n",$1);}

;

expr : expr'+'expr1 {$$=$1+$3;}

| expr'-'expr1 {$$=$1-$3;}

| expr'\*'expr1 {$$=$1\*$3;}

| expr'/'expr1 {$$=$1/$3;}

| expr1

;

expr1: '('expr')' {$$=$2;}

| DIGIT

;

%%

int main()

{

yyparse();

}

yyerror(char \*s)

{

printf("%s",s);

}

Testing:

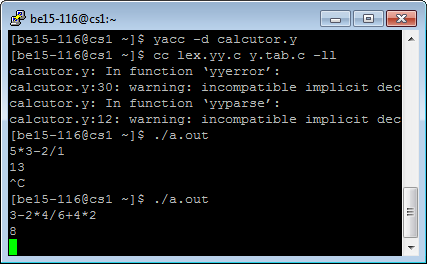
Input:

1+2\*3

Output:

7

Actual Output:



Result:

11. ***Progarm to find Follows for the given grammar.***

Source code:

#include<stdio.h>

main()

{

       int np,i,j,k;

       char prods[10][10],follow[10][10],Imad[10][10];

       printf("enter no. of productions\n");

       scanf("%d",&np);

       printf("enter grammar\n");

       for(i=0;i<np;i++)

       {

               scanf("%s",&prods[i]);

       }

       for(i=0; i<np; i++)

       {

               if(i==0)

               {

               printf("Follow(%c) = $\n",prods[0][0]);

               }

               for(j=3;prods[i][j]!='\0';j++)

               {

                    int temp2=j;

                    if(prods[i][j] >= 'A' && prods[i][j] <= 'Z')

                    {

                      if((strlen(prods[i])-1)==j)

                         {

                         printf("Follow(%c)=Follow(%c)\n",prods[i][j],prods[i][0]);

                         }

                         int temp=i;

                         char f=prods[i][j];

                         if(!isupper(prods[i][j+1])&&(prods[i][j+1]!='\0'))

                         printf("Follow(%c)=%c\n",f,prods[i][j+1]);

                         if(isupper(prods[i][j+1]))

                               {

                                repeat:

                                      for(k=0;k<np;k++)

                                      {

                                      if(prods[k][0]==prods[i][j+1])

                                      {

                                        if(!isupper(prods[k][3]))

                                        {

                                        printf("Follow(%c)=%c\n",f,prods[k][3]);

                                        }

                                        else

                                        {

                                        i=k;

                                        j=2;

                                        goto repeat;

                                        }

                                      }

                                      }

                               }

                               i=temp;

                       }

                       j=temp2;

               }

       }

}

Testing:

Input:

Enter no. of productions

3

Enter grammar

F->BC

B->b|c

C->d|e

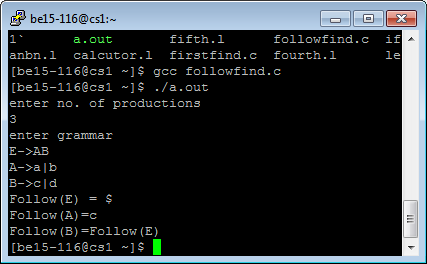
Output:

Follow(F)=$

Follow(B)=d

Follow(C)=Follow(F)

Actual Output:



Result:

10. ***Progarm to find FIRST for the given grammar.***

Source code:

#include<stdio.h>

#include<ctype.h>

int main()

{

       int i,n,j,k;

       char str[10][10],f;

       printf("Enter the number of productions\n");

       scanf("%d",&n);

       printf("Enter grammar\n");

       for(i=0;i<n;i++)

       scanf("%s",&str[i]);

       for(i=0;i<n;i++)

       {

               f= str[i][0];

               int temp=i;

               if(isupper(str[i][3]))

               {

               repeat:

                       for(k=0;k<n;k++)

                       {

                                if(str[k][0]==str[i][3])

                                {

                                       if(isupper(str[k][3]))

                                       {

                                               i=k;

                                               goto repeat;

                                       }

                                       else

                                       {

                                               printf("First(%c)=%c\n",f,str[k][3]);

                                       }

                                }

                       }

               }

               else

               {

                        printf("First(%c)=%c\n",f,str[i][3]);

               }

               i=temp;

       }

}

Testing:

Input:

Enter the number of profuctions

3

Enter grammar

A->B+C

B->b

C->c

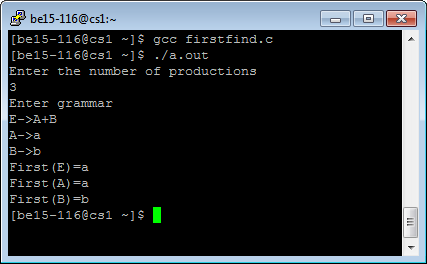
Output:

First(A)=b

First(B)=b

First(C)=c

Actual Output:



Result:

9. ***Program to implement stand alone scanner using LEX***

Source code:

%{

#include<stdio.h>

%}

identifier [a-zA-Z]+[\_]?[a-zA-Z0-9]\*

%%

#.\* {printf("\n %s PREPROCESSOR DIRECTIVE",yytext);}

auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|const|float|short|unsigned|continue|for|signed|void|default|goto|sizeof|volatile|do|if|static|while {printf("\n %s KEYWORD",yytext);}

{identifier} {printf("\n %s IDENTIFIER",yytext);}

"//" {printf("\n %s COMMENT",yytext);}

\".\*\" {printf("\n %s STRING",yytext);}

[{] {printf("\n %s BLOCk OPEN",yytext);}

[}] {printf("\n %s BLOCK CLOSE",yytext);}

[(] {printf("\n %s OPEN PARANTHESIS",yytext);}

[)] {printf("\n %s CLOSED PARANTHESIS",yytext);}

{identifier}\( {printf("\n %s FUNCTION",yytext);}

[\=] {printf("\n %s ASSIGNMENT OPERATOR",yytext);}

[-+\*/%<>] {printf("\n %s OPERATOR",yytext);}

{identifier}\\* {printf("\n %s PONITER",yytext);}

[;] {printf("\n %s DELIMITER",yytext);}

[+-]?[0-9]\* {printf("\n %s NUMBER",yytext);}

%%

void main()

{

yyin=fopen("program.c","r");

yylex();

}

**Testing:**

**Input:**

file.txt

#include<stdio.h>

int main()

{

   Int a, b, c;

printf("Enter two numbers to add\n");

   scanf("%d%d",&a,&b);

   c = a + b;

   printf("Sum of entered numbers = %d\n",c);

   return 0;

}

**Expected Output:**

Line    Token no        Token name      Lexeme

1       1               preprocessor directive  #include<stdio.h>

3       2               keyword         int

3       3               identifier      main

3       4               special char    (

3       5               special char    )

4       6               special char    {

5       7               keyword         int

5       8               identifier      a

5       9               special char    ,

5       10              identifier      b

5       11              special char    ,

5       12              identifier      c

5       13              terminator      ;

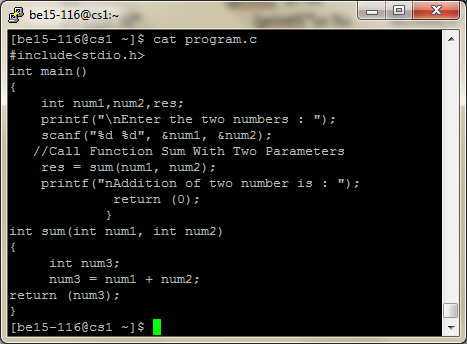
7       14              function        printf

7       15              special char    (

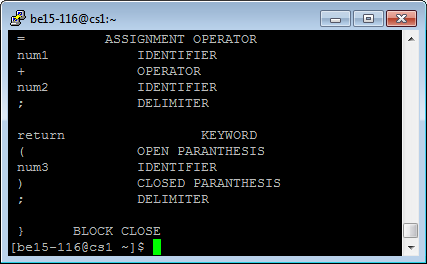
7       16              special char    "

7       17              identifier      Enter

Actual Input:



Actual Output:



Result:

8.Program to recognize real numbers using lex.

Source code:

%{

#include<stdio.h>

%}

%%

[+-]?[0-9]+[.]?[0-9]\* {printf("real number");}

%%

void main()

{

printf("enter a real number");

yylex();

}

Testing:

Input:

2.5

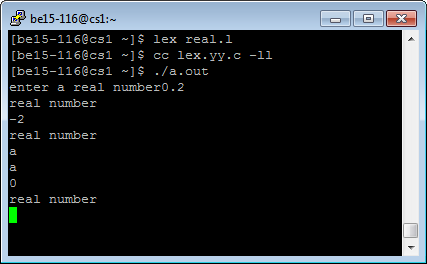
36

Expected Output:

2.5 is a floating number

36 is a integer

Actual Output:



Result:

7.Lex program to convert strings from upper to lower ,lower to upper case .

Source code:

%{

#include<stdio.h>

#include<ctype.h>

void to\_upper(char \*);

void to\_lower(char \*);

%}

lower [a-z]\*

upper [A-Z]\*

%%

{upper} to\_lower(yytext);

{lower} to\_upper(yytext);

[0-9a-zA-Z+-+\*!@#$%^{}()~`:;'\|<>,.\_]\* printf("invalid");

%%

void main()

{

printf("enter the string:\n");

yylex();

}

void to\_lower(char \*c)

{

int i=0;

for(i=0;c[i]!='\0';i++)

{

printf("%c",tolower(c[i]));

}

}

void to\_upper(char \*c)

{

int i=0;

for(i=0;c[i]!='\0';i++)

{

printf("%c",toupper(c[i]));

}

}

Testing:

Input:

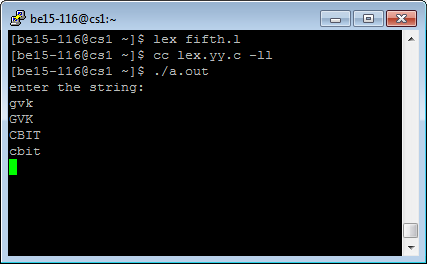
Enter the string:

cse

Output:

CSE

Actual Output:



Result:

6.lex program to recognize hexa,octa,decimal and invalid numbers.

Source code:

%{

#include<stdio.h>

%}

%%

[+-]?[0][0-7]+ {printf("octal number");}

[+-]?[0-9]+ {printf("decimal number");}

[+-]?[0][xX][0-9A-Fa-f]+ {printf("hexa number");}

[0-9a-zA-Z+\*=! .@#$%^&-]+ {printf("invalid");}

%%

void main()

{

printf("enter a number :\n");

yylex();

}

Testing:

Input:

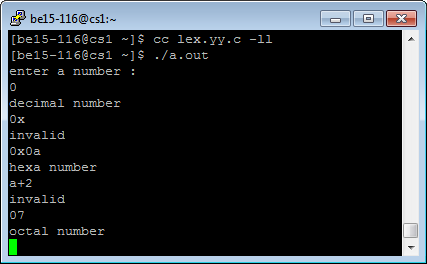
Enter a number:

06

Output:

invalid

Actual Output:



Result:

5.lex program to recognize a decimal number.

Source code:

%{

#include<stdio.h>

%}

%%

[0-9]\*[.][0-9]+ {printf("decimal number");}

%%

void main()

{

printf("enter a number\n");

yylex();

}

Testing:

Input:

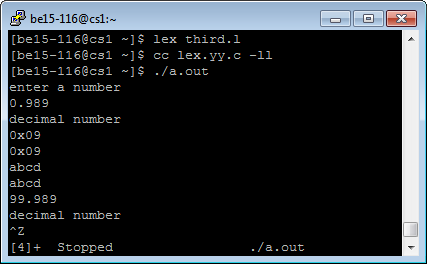
Enter a number

0.32

Output:

decimal number

Actual output:



Result:

4.lex program to recognize hexa and octal number.

Source code:

%{

#include<stdio.h>

%}

%%

[0][0-7]\* {printf("Octa number\t");}

[0][xX][0-9A-Fa-f]\* {printf("Hexanumber \t");}

%%

void main()

{

printf("Enter a number\n");

yylex();

}

Testing:

Input:

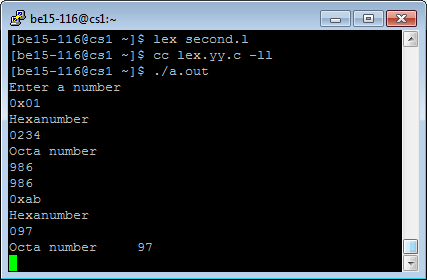
Enter a number

Ox21

Output:

Hexanumber

Actual Output:



Result:

3.lex program to print the strings which are upper and lowercase

Source code:

%{

#include<stdio.h>

int Upper=0;

int Lower=0;

%}

%%

[A-Z] {printf("Uppercase\t");Upper++;}

[a-z] {printf("Lowercase\t");Lower++;}

%%

void main()

{

printf("Enter a string\n");

yylex();

printf("Uppercase=%d and Lowercase=%d",Upper,Lower);

}

Testing:

Input:

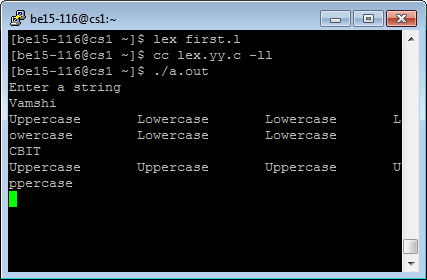
Enter a string:

Cbit

Output:

Uppercase lowercase lowercase lowercase

Actual Output:



Result:

2. ***Program to implement stand alone scanner in C***

Source code:

#include<stdio.h>

#include<ctype.h>

#include<string.h>

int main()

{

       FILE \*input, \*output;

       int l=1;

       int t=0;

       int j=0;

       int i,flag;

       char ch,str[20];

       input = fopen("input.txt","r");

       output = fopen("output.txt","w");

       char keyword[30][30] = {"int","main","if","else","do","while"};

       fprintf(output,"Line no.   \t Token no.          \t Token       \t Lexeme\n\n");

       while(!feof(input))

       {

               i=0;

               flag=0;

               ch=fgetc(input);

               if( ch=='+' || ch== '-' || ch=='\*' || ch=='/' )

               {

                       fprintf(output,"%7d\t\t %7d\t\t Operator\t %7c\n",l,t,ch);

                       t++;

               }

               else if( ch==';' || ch=='{' || ch=='}' || ch=='(' || ch==')' || ch=='?' || ch=='@' || ch=='!' || ch=='%')

               {

                       fprintf(output,"%7d\t\t %7d\t\t Special symbol\t %7c\n",l,t,ch);

                       t++;

               }

               else if(isdigit(ch))

               {

                       fprintf(output,"%7d\t\t %7d\t\t Digit\t\t %7c\n",l,t,ch);

                       t++;

               }

               else if(isalpha(ch))

               {

                       str[i]=ch;

                       i++;

                       ch=fgetc(input);

                       while(isalnum(ch) && ch!=' ')

                       {

                                str[i]=ch;

                                i++;

                               ch=fgetc(input);

                       }

                       str[i]='\0';

                       for(j=0;j<=5;j++)

                      {

                               if(strcmp(str,keyword[j])==0)

                               {

                                       flag=1;

                                       break;

                               }

                       }

                       if(flag==1)

                       {

                               fprintf(output,"%7d\t\t %7d\t\t Keyword\t %7s\n",l,t,str);

                               t++;

                       }

                       else

                       {

                               fprintf(output,"%7d\t\t %7d\t\t Identifier\t %7s\n",l,t,str);

                               t++;

                       }

               }

               else if(ch=='\n')

               {

                       l++;

               }

       }

       fclose(input);

       fclose(output);

       return 0;

}

**Testing:**

**Input:**

#include <stdio.h>

int main()

{

    int n, i;

   unsigned long long factorial = 1;

printf("Enter an integer: ");

    scanf("%d",&n);

// show error if the user enters a negative integer

    if (n < 0)

        printf("Error! Factorial of a negative number doesn't exist.");

else

    {

        for(i=1; i<=n; ++i)

        {

            factorial \*= i;              // factorial = factorial\*i;

        }

       printf("Factorial of %d = %llu", n, factorial);

   }

return 0;

}

**Expected Output:**

Line no.         Token no.               Token           Lexeme

     1                0                 Identifier      include

     1                1                 Identifier        stdio

     1                2                 Identifier            h

     2                3                 Keyword             int

     2                4                 Keyword            main

     2                5                 Special symbol        )

     3                6                 Special symbol        {

     4                7                 Keyword             int

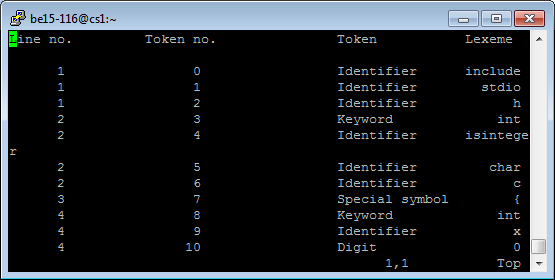
     4                8                 Identifier            n

     4                9                 Identifier            i

     5               10                 Identifier      unsigned

     5               11                 Identifier         long

Actual Output:



Result:

1. finding sum of integers,sum of floats,no. of words,integers,floats in a given text file.

Source code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

main()

{

FILE \*fp;

char ch, buff[200];

int lines, words, noi=0, nof=0, i=0, sign=0, temp=0, temp2=0, isum=0, flag\_int=1, fcount=0, flag\_char=0;

float fsum=0;

lines = 1;

words = 1;

fp = fopen("text.txt", "r");

if(fp == NULL)

perror("error while opening the file.\n");

while((ch = fgetc(fp)) != EOF)

{

buff[i] = ch;

i++;

if(ch=='\n'||ch=='\0')

{

++lines;

}

if(isspace(ch)||ch=='\t'||ch=='\n'||ch=='\0')

{

++words;

}

}

buff[i] = '\0';

printf("\n%s\n\n", buff);

for(i=0 ; buff[i]!='\0' ; i++)

{

printf("\nbuff = %c",buff[i]);

if((buff[i] == ' ') || (buff[i] == '\n'))

{

if(flag\_char == 1)

{

flag\_char=0;

printf("\nflag char = %d",flag\_char);

}

else

{

if(flag\_int !=0 )

{

noi++;

printf("\nnoi=%d",noi);

if(sign == 0)

{

isum = isum + temp;

printf("\nisum=%d",isum);

}

else

{

isum = isum - temp;

printf("\nisum=%d",isum);

}

}

if(flag\_int == 0)

{

nof++;

printf("\nnof=%d",nof);

if(sign == 0)

{

fsum = fsum + (temp + (temp2 \* (pow(10,-1\*fcount))));

printf("\nfsum=%f",fsum);

}

else

{

fsum = fsum - (temp + (temp2 \* (pow(10,-1\*fcount))));

printf("\nfsum=%f",fsum);

}

}

}

temp = 0;

temp2 = 0;

sign = 0;

flag\_int = 1;

fcount = 0;

printf("\nSet : temp=%d, temp2=%d, sign=%d, flag\_int=%d, fcount=%d",temp, temp2, sign, flag\_int, fcount);

}

else if(flag\_char == 1)

{

continue;

}

else if((buff[i]>='A' && buff[i]<='Z') || (buff[i]>='a' && buff[i]<='z'))

{

flag\_char = 1;

printf("\nflag char = %d",flag\_char);

continue;

}

else if(buff[i] == '+')

{

if((buff[i-1] == ' ') || (buff[i-1] == '\n'))

{

sign=0;

printf("\nsign=%d",sign);

}

else

{

flag\_char = 1;

printf("\nflag char = %d",flag\_char);

}

}

else if(buff[i] == '-')

{

if((buff[i-1] == ' ') || (buff[i-1] == '\n'))

{

sign=1;

printf("\nsign=%d",sign);

}

else

{

flag\_char = 1;

printf("\nflag char = %d",flag\_char);

}

}

else if(buff[i] == '.')

{

if(isdigit(buff[i-1]))

{

flag\_int = 0;

printf("\nflag int = %d",flag\_int);

}

else

{

flag\_char = 1;

printf("\nflag char = %d",flag\_char);

}

}

else if(isdigit(buff[i]))

{

if(flag\_int !=0 )

{

temp = temp\*10 + (buff[i] - '0');

printf("\ntemp=%d",temp);

}

else

{

fcount++;

printf("\nfcount=%d",fcount);

temp2 = temp2\*10 + (buff[i] - '0');

printf("\ntemp2=%d",temp2);

}

}

else

{}

}

if(temp!=0)

{

printf("\ntemp=%d",temp);

if(flag\_int !=0 )

{

noi++;

printf("\nnoi=%d",noi);

if(sign == 0)

{

isum = isum + temp;

printf("\nisum=%d",isum);

}

else

{

isum = isum - temp;

printf("\nisum=%d",isum);

}

}

if(flag\_int == 0)

{

nof++;

printf("\nnof=%d",nof);

if(sign == 0)

{

fsum = fsum + (temp + (temp2 \* (pow(10,-1\*fcount))));

printf("\nfsum=%d",fsum);

}

else

{

fsum = fsum - (temp + (temp2 \* (pow(10,-1\*fcount))));

printf("\nfsum=%d",fsum);

}

}

}

printf("lines = %d\nwords = %d\nintegers = %d\nfloats = %d\n\nSum of integers = %d\nSum of floats = %f\n",lines,words,noi,nof,isum,fsum);

}

**Testing:**

**Input:**

cbit 4.4.4

+3.2 -2.3 3

++2.34 -3 22

**Expected Output:**

Number of words:8

Number of lines:3

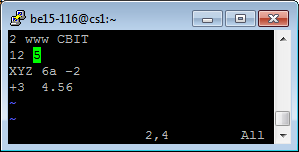
Number of integers:3

Number of floats:2

Sum of integers:22

Sum of floats:0.9

Actual Input:



Actual Output:

Result: