Alaparthi shasank sai

AP20110010323

Compiler Design

WEEK 1

**1.**

#include<stdio.h>

int main()

{

int state=0,i=0;

char token,input[20];

printf("Enter input string \t :");

scanf("%s",input);

while((token=input[i++])!='\0')

{

switch(state)

{

case 0: if(token=='0')

state=1;

else if(token=='1')

state=2;

else

{

printf("Invalid token");

}

break;

case 1: if(token=='0')

state=0;

else if(token=='1')

state=3;

else

{

printf("Invalid token");

}

break;

case 2: if(token=='0')

state=3;

else if(token=='1')

state=0;

else

{

printf("Invalid token");

}

break;

case 3: if(token=='0')

state=2;

else if(token=='1')

state=1;

else

{

printf("Invalid token");

}

break;

}

}

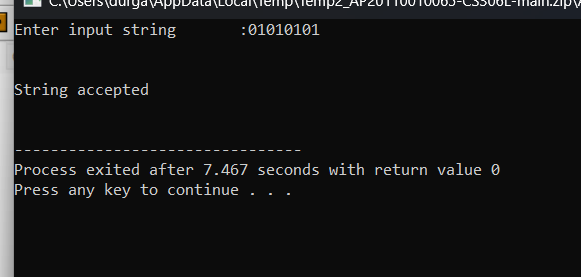
if(state==0)

printf("\n\nString accepted\n\n");

else

printf("\n\nString not accepted\n\n");

}



**1b.**

**CODE**

#include <stdio.h>

#include<stdlib.h>

int main()

{

printf("Enter the input string: ");

char ch , inp[20];//input string

scanf("%s",inp);

int i=0,state=1;

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

{

ch=inp[i];

switch(state)

{

case 1:if(ch=='a')

{

state=2;

}

else if(ch=='b')

{

state=4;

}

else

{

printf("invalid token");

exit(0);

}

break;

case 2:if(ch=='a')

{

state=3;

}

else if(ch=='b')

{

state=4;

}

else

{

printf("invalid token");

exit(0);

}

break;

case 3:if(ch=='a')

{

state=3;

}

else if(ch=='b')

{

state=4;

}

else

{

printf("invalid token");

exit(0);

}

break;

case 4:if(ch=='a')

{

state=2;

}

else if(ch=='b')

{

state=5;

}

else

{

printf("invalid token");

exit(0);

}

break;

case 5:if(ch=='a')

{

state=2;

}

else if(ch=='b')

{

state=5;

}

else

{

printf("invalid token");

exit(0);

}

break;

}

i++;

}

if(state==3||state==5)

{

printf(" String is accepted");

}

else

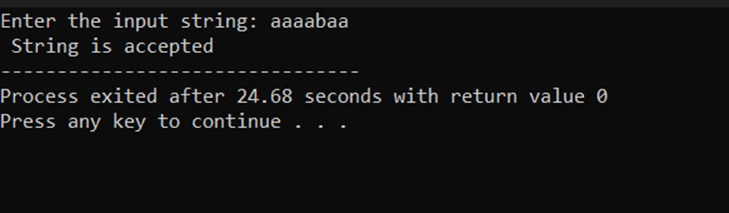
{

printf(" String is not accepted");

}

return 0;

}



WEEK 2

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <string.h>

#include <ctype.h>

int size = 0;

char keyword[30][30] = {"int", "while", "break", "for", "do", "if", "float", "char", "switch", "double", "short", "long", "unsigned", "sizeof", "else", "register", "extern", "static", "auto", "case", "break", "volatile", "enum", "typedef"};

char id[20], num[10], op[5];

char symb\_tab[50][20];

int idx = 0;

int check\_keyword(char s[])

{

int i;

for (i = 0; i < 24; i++)

if (strcmp(s, keyword[i]) == 0)

return 1;

return 0;

}

void store\_symb\_tab(char id[], char symb\_tab[][20])

{

int flag = 0, i;

for (i = 0; i < idx; i++)

{

if (strcmp(id, symb\_tab[i]) == 0)

{

flag = 1;

}

}

if (flag == 0)

{

strcpy(symb\_tab[idx], id);

idx++;

}

}

void main()

{

FILE \*fp1, \*fp2;

char c;

int state = 0;

int i = 0, j = 0, k = 0;

fp1 = fopen("Input.txt", "r");

fp2 = fopen("Output.txt", "w");

while ((c = fgetc(fp1)) != EOF)

{

switch (state)

{

case 0:

if (isalpha(c))

{

state = 1;

id[i++] = c;

}

else if (isdigit(c))

{

state = 3;

num[j++] = c;

}

else if (c == '<' || c == '>')

{

state = 5;

op[k++] = c;

}

else if (c == '=' || c == '!')

{

state = 8;

op[k++] = c;

}

else if (c == '/')

state = 10;

else if (c == ' ' || c == '\t' || c == '\n')

state = 0;

else

fprintf(fp2, "\n%c", c);

break;

case 1:

if (isalnum(c))

{

state = 1;

id[i++] = c;

}

else

{

id[i] = '\0';

if (check\_keyword(id))

fprintf(fp2, "\n%s : keyword ", id);

else

{

fprintf(fp2, "\n%s : identifier", id);

store\_symb\_tab(id, symb\_tab);

}

state = 0;

i = 0;

ungetc(c, fp1);

}

break;

case 3:

if (isdigit(c))

{

num[j++] = c;

state = 3;

}

else

{

num[j] = '\0';

fprintf(fp2, "\n%s: number", num);

state = 0;

j = 0;

ungetc(c, fp1);

}

break;

case 5:

if (c == '=')

{

state = 0;

}

else

{

state = 0;

ungetc(c, fp1);

}

op[k++] = c;

op[k] = '\0';

fprintf(fp2, "\n%s : relational operator ", op);

k = 0;

break;

case 8:

if (c == '=')

{

op[k++] = c;

op[k] = '\0';

fprintf(fp2, "\n%s : relational operator ", op);

k = 0;

state = 0;

}

else

{

ungetc(c, fp1);

state = 0;

}

break;

case 10:

if (c == '\*')

state = 11;

else

fprintf(fp2, "\ninvalid lexeme");

break;

case 11:

if (c == '\*')

state = 12;

else

state = 11;

break;

case 12:

if (c == '\*')

state = 12;

else if (c == '/')

state = 0;

else

state = 11;

break;

}

}

if (state == 11)

fprintf(fp2, "comment did not close");

fclose(fp1);

fclose(fp2);

}

**Week-3**

1. **Identification of Vowels and Consonants**

%option noyywrap

%{

#include<stdio.h>

%}

%%

[aeiouAEIOU]+ {printf("vowel\n");}

[a-zA-Z] {printf("consonant\n");}

%%

int main()

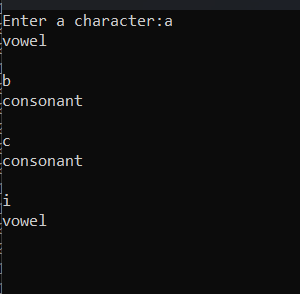
{

printf("Enter a character:");

yylex();

return 0;

}



**b. count number of vowels and consonants**

**code:**

%option noyywrap

%{

#include<stdio.h>

int v\_c = 0;

int c\_c = 0;

%}

%%

[aeiouAEIOU] {v\_c++;}

[a-zA-Z] {c\_c++;}

%%

int main()

{

printf("Enter string of vowels and consonants:");

yylex();

printf("Number of vowels: %d\n",v\_c);

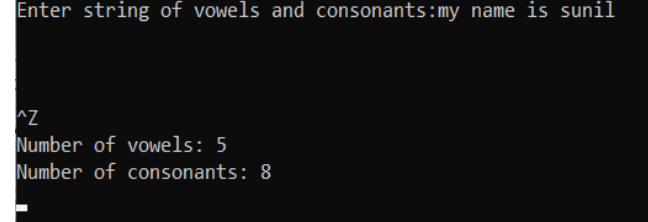
printf("Number of consonants: %d\n",c\_c);

getch();

return 0;

}

**Output**

****

**c. Count the number of Lines in given input**

**code:**

%option noyywrap

%{

#include<stdio.h>

int c=0;

%}

%%

\n c++;

.

%%

int main()

{

printf("Enter Input:");

yylex();

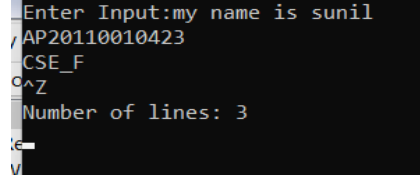
printf("Number of lines: %d\n",c);

getch();

return 0;

}

**Output:**

****

**d. Recognize strings ending with 00**

**code:**

%option noyywrap

%{

#include<stdio.h>

%}

%%

[0-1]\*00 {printf("Given string ends with 00");}

[0-1]\* {printf("Given string does not ends with 00");}

%%

int main()

{

printf("Enter the string:");

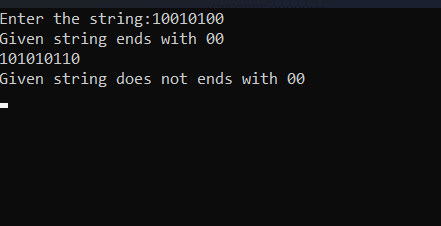
yylex();

getch();

return 0;

}

**Output:**



**e. Recognize a string with three consecutive 0’s**

**code:**

%option noyywrap

%{

#include<stdio.h>

%}

%%

[0-1]\*000[0-1]\* {printf("Given string has three

consecutive zeros");}

[0-1]\* {printf("Given string does not have three

consecutive zeros");}

%%

int main()

{

printf("Enter the string:");

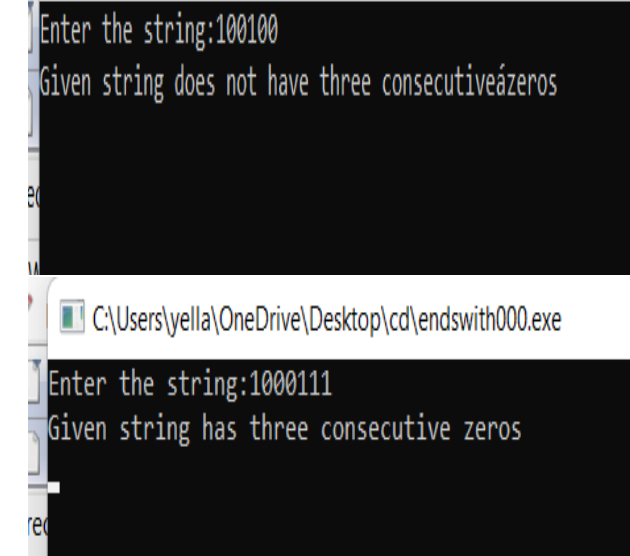
yylex();

getch();

return 0;

}

**Output**

****

WEEK 4

%%

auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|continue|for|signed|void|do|if|static|while|default|goto|sizeof|volatile|const|float|short printf("Is a keyword");

[a-zA-Z]([a-zA-Z]|[0-9])\* printf("Is an identifier");

[0-9]+ printf("Is an integer");

[<|>|<=|>=|==|!=] printf("Is a Relational Operator");

.\* printf("Invalid Expression");

%%

int yywrap(void)

{

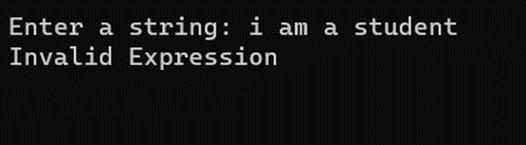
}

main() {

printf("Enter a string: ");

  yylex();

}



Week 6: Recursive Descent Parser

6.

//grammar

/\*

E->TE'

E'->+TE'|NULL

T->FT'

T'->\*FT'|NULL

F->(E)|a

\*/

#include<stdio.h>

#include<string.h>

void E();

void T();

void F();

void Eds();

void Tds();

char input[1000];

int i,error;

int main()

{

printf("Enter the input:\n");

gets(input);

E();

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

{

printf("String is accepted");

}

else

{

printf("string is rejected");

}

}

void E()

{

T();

Eds();

}

void T()

{

F();

Tds();

}

void Eds()

{

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

{

i++;

T();

Eds();

}

}

void Tds()

{

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

{

i++;

F();

Tds();

}

}

void F()

{

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

{

i++;

}

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

{

i++;

E();

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

{

i++;

}

else

{

error=1;

}

}

else

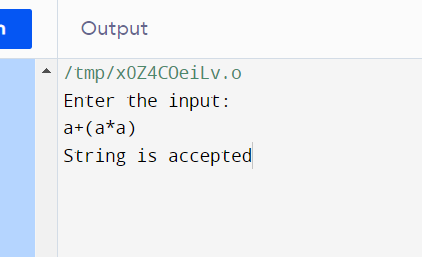
{

error=1;

}

}

OUTPUT:



7.

#include<stdio.h>

#include<string.h>

int S() ,L() ,Ldash();

char \*ip;

char string[50];

int main()

{

printf("Enter the string\n");

scanf("%s",string);

ip=string;

printf("\n\nInput\tAction\n-----------------------\n");

if(S() && \*ip=='\0'){

printf("\n-----------------------\n");

printf("\n String is accepted\n");

}

else{

printf("\n---------------------------\n");

printf("String not accepted\n");

}

}

int S()

{

if(\*ip=='(')

{

printf("%s\tS->(L) \n",ip);

ip++;

if(L())

{

if(\*ip==')')

{

ip++;

return 1;

}

else

return 0;

}

else

return 0;

}

else if(\*ip=='a')

{

ip++;

printf("%s\tS->a \n",ip);

return 1;

}

else

return 1;

}

int L(){

printf("%s\tL->SL' \n",ip);

if(S())

{

if(Ldash())

{

return 1;

}

else

return 0;

}

else

return 0;

}

int Ldash(){

if(\*ip==',')

{

printf("%s\tL'->,SL' \n",ip);

ip++;

if(S())

{

if(Ldash()){

return 1;

}

else

return 0;

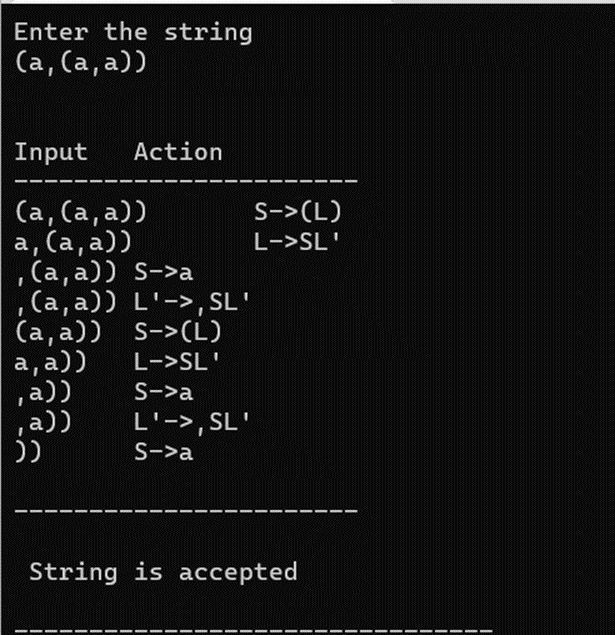
}

else

return 0;

    }

}



**WEEK-7**

FIND FIRST AND FOLLOW

#include<stdio.h>

#include<ctype.h>

#include<string.h>

#include<math.h>

#include<stdlib.h>

int n,m=0,p,i=0,j=0;

char a[10][10],f[10];

void follow(char c);

void first(char c);

int main()

{

int i,z;

char c,ch;

printf("Enter the no of productions:\n");

scanf("%d",&n);

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

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

{

scanf("%s%c",a[i],&ch);

}

do

{

m=0;

printf("Enter the elements whose first&follow is to be found:");

scanf("%c",&c);

first(c);

printf("First(%c)={",c);

for(i=0;i<m;i++)

{

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

}

printf("}\n");

strcpy(f," ");

m=0;

follow(c);

printf("Follow(%c)={",c);

for(i=0;i<m;i++)

{

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

}

printf("}\n");

printf("continue(0/1)?");

scanf("%d%c",&z,&ch);

}

while(z==1);

return(0);

}

void first(char c)

{

int k;

if(!isupper(c))

{

f[m++]=c;

}

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

{

if(a[k][0]==c)

{

if(islower(a[k][2]))

f[m++]=a[k][2];

else

{

first(a[k][2]);

}

}

}

}

void follow(char c)

{

if(a[0][0]==c)

{

f[m++]='$';

}

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

{

for(j=2;j<strlen(a[i]);j++)

{

if(a[i][j]==c)

{

if(a[i][j+1]!='\0')

first(a[i][j+1]);

if(a[i][j+1]=='\0'&&c!=a[i][0])

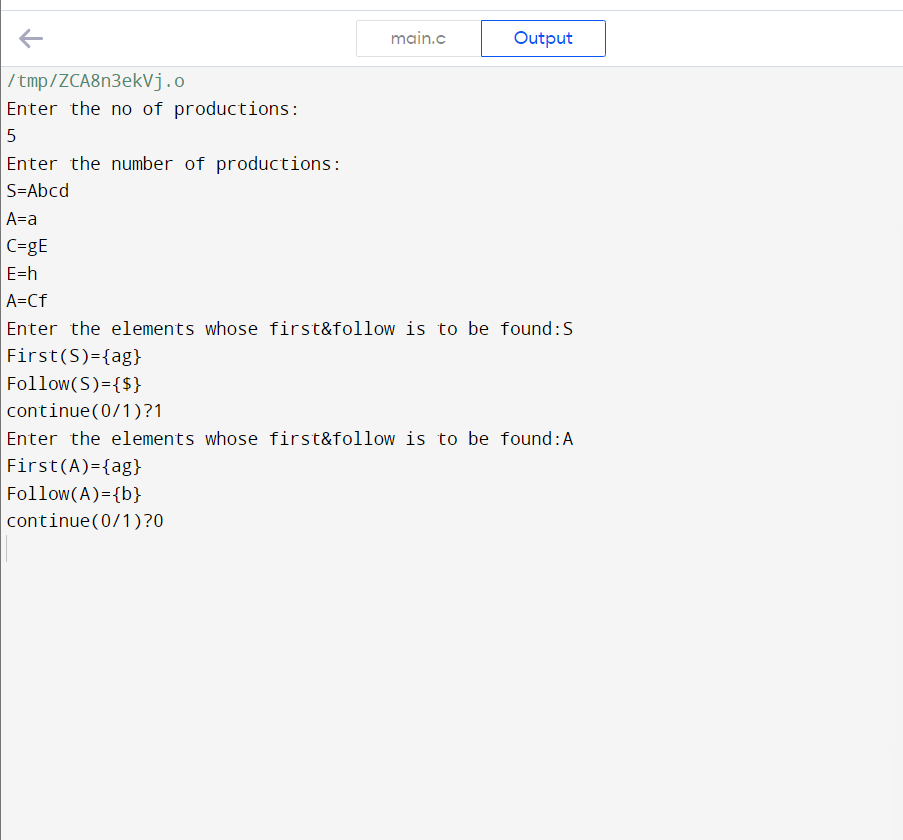
follow(a[i][0]);

}

}

}

}



W8

PREDECIVE PARSER:

9

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

int i=0,top=0;

char stack[20],ip[20];

void push(char c)

{

if (top>=20)

printf("Stack Overflow");

else

stack[top++]=c;

}

void pop(void)

{

if(top<0)

printf("Stack underflow");

else

top--;

}

void error(void)

{

printf("\n\nSyntax Error!!!! String is invalid\n");

getch();

exit(0);

}

int main()

{

int n;

printf("The given grammar is\n\n");

printf("S -> aBa\n");

printf("B -> bB | epsilon \n\n");

printf("Enter the string to be parsed:\n");

scanf("%s",ip);

n=strlen(ip);

ip[n]='$';

ip[n+1]='\0';

push('$');

push('S');

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

{ if(ip[i]=='$' && stack[top-1]=='$')

{

printf("\n\n Successful parsing of string \n");

return(1);

}

else

if(ip[i]==stack[top-1])

{

printf("\nmatch of %c occured ",ip[i]);

i++;pop();

}

else

{

if(stack[top-1]=='S' && ip[i]=='a')

{

printf(" \n S ->aBa");

pop();

push('a');

push('B');

push('a');

}

else

if(stack[top-1]=='B' && ip[i]=='b')

{

printf("\n B ->bB");

pop();push('B');push('b');

}

else

if(stack[top-1]=='B' && ip[i]=='a')

{

printf("\n B -> epsilon");

pop();

}

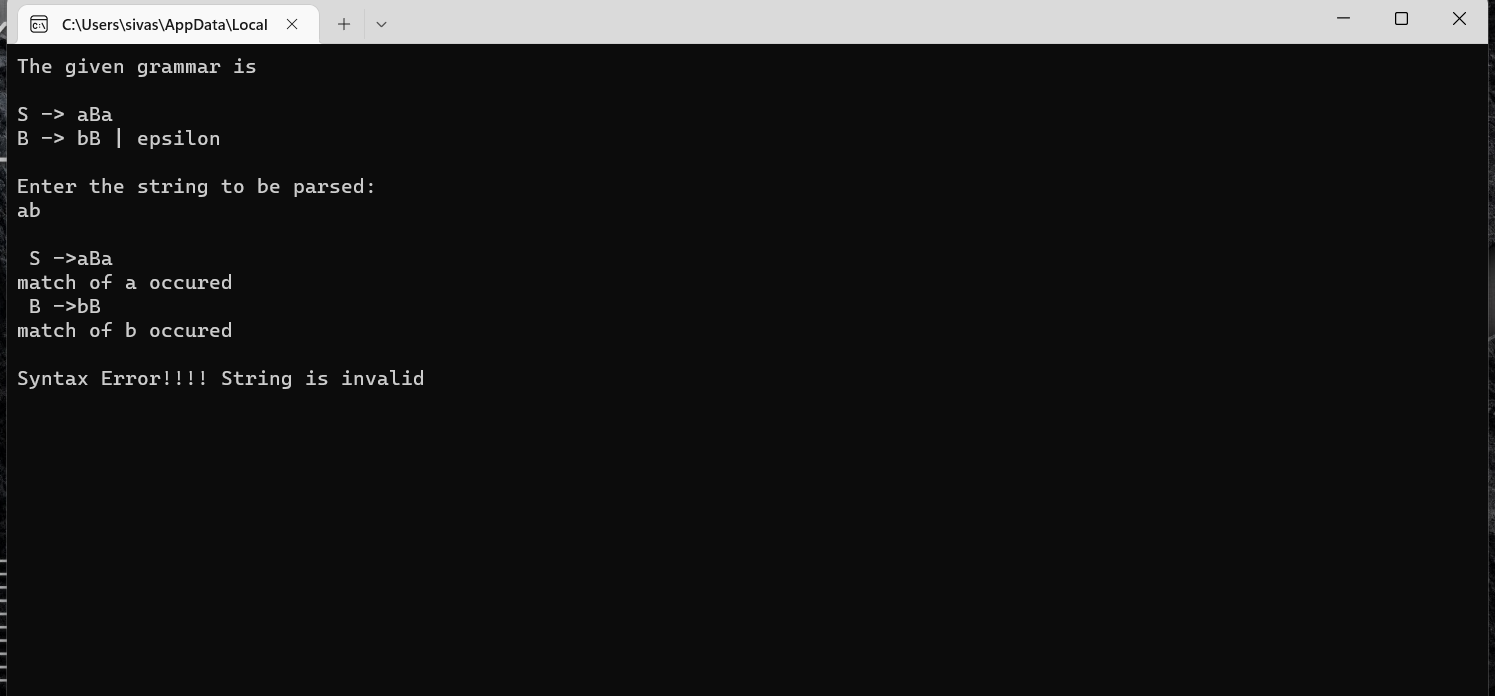
else

error();

}

}

}//end of main



10

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

int i=0,top=0;

char stack[20],ip[20];

void push(char c)

{

if(top>=20)

{

printf("Stack Overflow");

}

else

{

stack[top++]=c;

}

}

void pop(void)

{

if(top<0)

{

printf("stack underflow");

}

else

{

top--;

}

}

void error(void)

{

printf("\n syntax error");

getch();

exit(0);

}

int main()

{

int n;

printf("The given grammar is\n\n");

printf("E->TE'\n");

printf("E'->+TE'|epsilon\n");

printf("T->FT'\n");

printf("T'->\*FT'|epsilon\n");

printf("F->(E)|d\n\n");

printf("Enter the string to be parsed:\n");

scanf("%s",ip);

n=strlen(ip);

ip[n]='$';

ip[n+1]='\0';

push('$');

push('E');

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

{

if(ip[i]=='$'&&stack[top-1]=='$')

{

printf("\n\n successful parsing of string\n");

return 1;

}

else if(ip[i]==stack[top-1])

{

printf("\nMatch of %c",ip[i]);

i++;

pop();

}

else

{

if(stack[top-1]=='E'&&(ip[i]=='d'||ip[i]=='('))

{

printf("\n E->TE'");

pop();

push('A');

push('T');

}

else if(stack[top-1]=='A'&&ip[i]=='+')

{

printf("\nE'->+TE'");

pop();

push('A');

push('T');

push('+');

}

else if(stack[top-1]=='A'&&(ip[i]==')'||ip[i]=='$'))

{

printf("E'->epsilon");

pop();

}

else if(stack[top-1]=='T'&&(ip[i]=='d'||ip[i]=='('))

{

printf("\n T->FT'");

pop();

push('B');

push('F');

}

else if(stack[top-1]=='B'&&ip[i]=='\*')

{

printf("\n T->\*FT'");

pop();

push('B');

push('F');

push('\*');

}

else if(stack[top-1]=='B'&&(ip[i]=='+'||ip[i]==')'||ip[i]=='$'))

{

printf("\nT'->epsilon");

pop();

}

else if(stack[top-1]=='F'&&ip[i]=='d')

{

printf("\n F->d");

pop();

push('d');

}

else if(stack[top-1]=='F'&&ip[i]=='(')

{

printf("\n F->(E)");

pop();

push(')');

push('E');

push('(');

}

else

{

error();

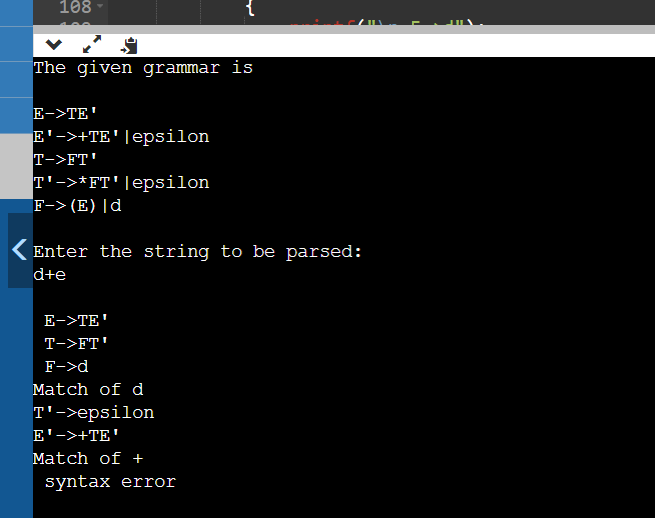
}

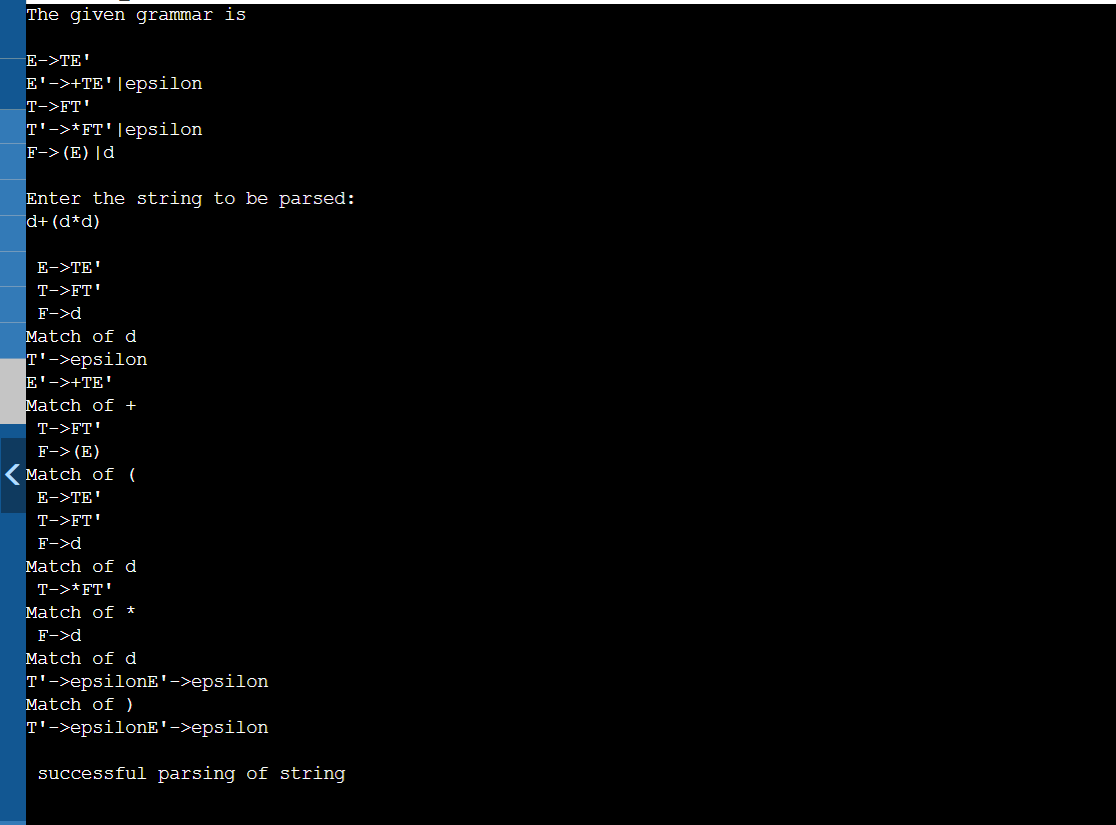
}

}

}

OUTPUT:





W9

11.SHIFT REDUCE PARSER:

#include<stdio.h>

#include<stdlib.h>

void pop(),push(char),display();

char stack[100]="\0";

char inputbuffer[100];

int top=-1;

char \*ip;

void main()

{

printf("E->E+E\n");

printf("E\_>E\*E\n");

printf("E->(E)\n");

printf("E->d\n");

printf("Enter the input string followed by $\n");

scanf("%s",inputbuffer);

ip=inputbuffer;

push('$');

printf("STACK\tBUFFER\tACTION\n");

printf("----\t----\t----\n");

display();

do

{

if((stack[top]=='E'&&stack[top-1]=='$')&&(\*(ip)=='$'))

break;

if(stack[top]=='$')

{

push(\*ip);

ip++;

printf("shift");

}

else if(stack[top]=='d')

{

display();

pop();

push('E');

printf("Reduce E->d\n");

}

else if(stack[top]=='E'&&stack[top-1]=='+'&&stack[top-2]=='E'&&ip!=' ')

{

display();

pop();

pop();

pop();

push('E');

printf("Reduce E->E+E");

}

else if(stack[top]=='E'&&stack[top-1]=='\*'&&stack[top-2]=='E')

{

display();

pop();

pop();

pop();

push('E');

printf("Reduce E->E\*E");

}

else if(stack[top]==')'&&stack[top-1]=='E'&&stack[top-2]=='(')

{

display();

pop();

pop();

pop();

push('E');

printf("Reduce E->(E)");

}

else

{

display();

push(\*ip);

ip++;

printf("shift");

}

}

while(1);

display();

printf("Accept\n\n\n");

}

void push(char c)

{

top++;

stack[top]=c;

}

void pop()

{

stack[top]='\0';

top--;

}

void display()

{

printf("\n%s\t%s\t",stack,ip);

}

OUTPUT:



12

#include<stdio.h>

#include<stdlib.h>

void pop(),push(char),display();

char stack[100]="\0", input[100], \*ip;

int top=-1;

void push(char c)

{

top++;

stack[top]=c;

}

void pop()

{

stack[top]='\0';

top--;

}

void display()

{

printf("\n%s\t%s\t",stack,ip);

}

void main()

{

printf("S->0S0\n");

printf("S->1S1\n");

printf("S->2\n");

printf("Enter the input string followed by $ \n");

scanf("%s",input);

ip=input;

push('$');

printf("STACK\t BUFFER \t ACTION\n");

printf("-----\t ------- \t------ \n");

display();

if(stack[top]=='$' && \*ip=='$'){

printf("Null Input");

exit(0);

}

do

{

if((stack[top]=='S' && stack[top-1]=='$') && (\*(ip)=='$'))

{

display();

printf(" Valid\n\n\n");

break;

}

if(stack[top]=='$')

{

push(\*ip);

ip++;

printf("Shift");

}

else if(stack[top]=='2')

{

display();

pop();

push('S');

printf("Reduce S->2");

}

else if(stack[top]=='0' && stack[top-1]=='S' && stack[top-2]=='0')

{

display();

pop();

pop();

pop();

push('S');

printf("Reduce S->0S0");

}

else if(stack[top]=='1' && stack[top-1]=='S' && stack[top-2]=='1')

{

display();

pop();

pop();

pop();

push('S');

printf("Reduce S->1S1");

}

else if(\*ip=='$')

{ printf(" Invalid\n\n\n");

break;

}

else

{

display();

push(\*ip);

ip++;

printf("shift");

}

}while(1);

}

WEEK 10

13.

%{

#include<stdio.h>

%}

%token NUMBER

%%

S: E { printf("The result is =%d\n",$1);}

;

E: E'+'T { $$ = $1 + $3; }

| T { $$ = $1;}

;

T: T'\*'F { $$ = $1 \* $3; }

| F { $$ = $1;}

;

F: '('E')' { $$ = $2;}

| NUMBER { $$ = $1;}

;

%%

int main(){

yyparse();

}

int yywrap(){

return 1;

}

void yyerror(char \*s){

printf("Error %s",s);

}

14

%{

#include<stdio.h>

%}

%token DIGIT

%left '-' '+'

%left '\*' '/'

%nonassoc UMINUS

%%

S: E { printf("The result is =%d\n",$1);}

;

E: E'+'E { $$ = $1 + $3; }

| E'-'E { $$ = $1 - $3; }

| E'\*'E { $$ = $1 \* $3; }

| E'/'E { if($3 == 0)

yyerror("Divide by zero");

else

$$ = $1 / $3; }

| '-'E %prec UMINUS { $$ = -$2; }

| '(' E ')' { $$ = $2; }

| DIGIT {$$ = $1;}

;

%%

int main()

{

yyparse();

}

int yywrap(){

return 1;

}

void yyerror(char \*s){

printf("Error %s",s);exit(0);

}

**Week-12**

Generate 3-address code for while statement using LEX and YACC.

Code:

Whilef.l Code:

%%

while return WHILE;

[A-Za-z]([A-Za-z]|[0-9])\* return ID;

[0-9]+ {return NUM;}

[ \t] ;

\n yyterminate();

. return yytext[0];

%%

Whilef.y Code:

%token ID NUM WHILE

%right '='

%left '+' '-'

%left '\*' '/'

%left MINUS

%%

S : WHILE{L1();} '(' E ')' {Lcond();} E ';' {End();}

E :V '='{push();} E{codegen\_assign();}

| E '+'{push();} E{codegen\_assign();}

| E '-'{push();} E{codegen\_assign();}

| E '\*'{push();} E{codegen\_assign();}

| E '/'{push();} E{codegen\_assign();}

| '(' E ')'

| '-'{push();} E{codegen\_assign();} %prec MINUS

| V

| NUM{push();}

;

V : ID {push();}

;

%%

#include "lex.yy.c"

#include<stdio.h>

char st[100][10];

int top=0;

char temp[3]="t0";

main()

{

printf("Enter the expression : ");

yyparse();

}

push()

{

strcpy(st[++top],yytext);

}

codegen()

{

printf("%s = %s %s %s\n", temp, st[top-2], st[top-1],st[top]);

top-=2;

strcpy(st[top],temp);

temp[1]++;

}

codegen\_umin()

{

printf("%s = -%s\n", temp, st[top]);

top--;

strcpy(st[top],temp);

temp[1]++;

}

codegen\_assign()

{

printf("%s = %s\n", st[top-2],st[top]);

top-=2;

}

L1(0)

{

printf("\nL1: \n");

}

Lcond()

{

printf("%s = not %s\n", temp,st[top]);

printf("if %s goto End\n", temp);

temp[1]++;

}

End()

{

printf("goto L1\n");

printf("End: end of while loop \n\n");

}

**Week-13**

|  |
| --- |
| #include<stdio.h> |
|  |

|  |
| --- |
| #include<string.h> |
|  |

|  |
| --- |
| #include<ctype.h> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| void input(); |
|  |

|  |
| --- |
| void output(); |
|  |

|  |
| --- |
| void change(int p,char \*res); |
|  |

|  |
| --- |
| void constant(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| struct expr |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| char op[2],op1[5],op2[5],res[5]; |
|  |

|  |
| --- |
| int flag; |
|  |

|  |
| --- |
| }arr[10]; |
|  |

|  |
| --- |
| int n; |
|  |

|  |
| --- |
| void main() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| input(); |
|  |

|  |
| --- |
| constant(); |
|  |

|  |
| --- |
| output(); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| void input() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| int i; |
|  |

|  |
| --- |
| printf("\n\nEnter the maximum number of expressions : "); |
|  |

|  |
| --- |
| scanf("%d",&n); |
|  |

|  |
| --- |
| printf("\nEnter the input : \n"); |
|  |

|  |
| --- |
| for(i=0;i<n;i++) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| scanf("%s",arr[i].op); |
|  |

|  |
| --- |
| scanf("%s",arr[i].op1); |
|  |

|  |
| --- |
| scanf("%s",arr[i].op2); |
|  |

|  |
| --- |
| scanf("%s",arr[i].res); |
|  |

|  |
| --- |
| arr[i].flag=0; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| void constant() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| int i; |
|  |

|  |
| --- |
| int op1,op2,res; |
|  |

|  |
| --- |
| char op,res1[5]; |
|  |

|  |
| --- |
| for(i=0;i<n;i++) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| if(isdigit(arr[i].op1[0]) && isdigit(arr[i].op2[0]) || strcmp(arr[i].op,"=")==0) |
|  |

|  |
| --- |
| /\*if both digits, store them in variables\*/ |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| op1=atoi(arr[i].op1); |
|  |

|  |
| --- |
| op2=atoi(arr[i].op2); |
|  |

|  |
| --- |
| op=arr[i].op[0]; |
|  |

|  |
| --- |
| switch(op) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| case '+': |
|  |

|  |
| --- |
| res=op1+op2; |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case '-': |
|  |

|  |
| --- |
| res=op1-op2; |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case '\*': |
|  |

|  |
| --- |
| res=op1\*op2; |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case '/': |
|  |

|  |
| --- |
| res=op1/op2; |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| case '=': |
|  |

|  |
| --- |
| res=op1; |
|  |

|  |
| --- |
| break; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| sprintf(res1,"%d",res); |
|  |

|  |
| --- |
| arr[i].flag=1; |
|  |

|  |
| --- |
| change(i,res1); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| void output() |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| int i=0; |
|  |

|  |
| --- |
| printf("\nOptimized code is : "); |
|  |

|  |
| --- |
| for(i=0;i<n;i++) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| if(!arr[i].flag) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| printf("\n%s %s %s %s",arr[i].op,arr[i].op1,arr[i].op2,arr[i].res); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| void change(int p,char \*res) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| int i; |
|  |

|  |
| --- |
| for(i=p+1;i<n;i++) |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| if(strcmp(arr[p].res,arr[i].op1)==0) |
|  |

|  |
| --- |
| strcpy(arr[i].op1,res); |
|  |

|  |
| --- |
| else if(strcmp(arr[p].res,arr[i].op2)==0) |
|  |

|  |
| --- |
| strcpy(arr[i].op2,res); |

|  |
| --- |
|  |

**Week-14**

import java.io.\*;

import java.util.\*;

class subexp\_opt

{

public static void main(String args[])throws IOException

{

String s,temp;

String arr[][]=new String[10][2]; //assuming 10 unique operations with LHS and RHS stored

int flag=0,index=0;

BufferedReader br=new BufferedReader(new InputStreamReader(new FileInputStream("input.txt")));

File op = new File("output.txt");

if (!op.exists())

op.createNewFile();

BufferedWriter output = new BufferedWriter(new FileWriter(op.getAbsoluteFile()));

for(;(s=br.readLine())!=null;flag=0)

{

temp=s.substring(s.indexOf("=")+1);

for(int i=0;i<index;i++)

{

if(temp.equals(arr[i][1]))

{

flag=1;

break;

}

else if(temp.contains(arr[i][1]))

s=s.replaceAll(arr[i][1],arr[i][0]);

}

if(flag==0)

{

arr[index][0]=s.substring(0,s.indexOf("="));

arr[index][1]=temp;

index++;

output.write(s);

output.newLine();

}

}

output.close();

}

}