Program – Top Down Parser

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#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();
int main(){
   i=0;error=0;
   printf("Enter an arithmetic expression : " );
   gets(input);
   E();
   if(strlen(input)==i && (error==0))
      printf("Accepted\n");
   else
      printf("Rejected\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

```
Enter an arithmetic expression : 1+2*3
Accepted
Enter an arithmetic expression : 1-2*3
Rejected
Enter an arithmetic expression : 1*+3
Rejected
```

Program – Lexical Analyzer

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int operator_checker(char op);
int special_checker(char sp);
int keyword_checker(char keyword[]);
int main()
{
   FILE *fp,*output;
   char
buffer[1000],ch,digit[100],identifier[100];
   int n=0,l=1,t=1,m=0;
   fp=fopen("input.txt","r");
   output=fopen("output.txt","w");
   fprintf(output,"Line no. \t Token no. \t
Token \t\t Lexeme\n\n");
   while(!feof(fp)) {
      buffer[n++]=fgetc(fp);
   }
   printf("\nInput file contents are :
\n%s",buffer);
   printf("\n");
   int len=strlen(buffer);
   while(m<len){
      ch=buffer[m];
      if(ch=='\n'){
         l++;
      }
      if(operator_checker(ch)==1)
      {
         fprintf(output,"%7d\t\t
%7d\tOperator\t%7c\n",l,t,ch);
         t++;
         m++;
         continue;
      }
      if(special_checker(ch)==1){
         fprintf(output,"%7d\t\t %7d\tSpecial
symbol\t%7c\n",l,t,ch);
         t++;
         m++;
         continue;
      }
      if(isdigit(ch)){
         int y=0;
         while(isdigit(ch)){
            if(ch==' ' || ch==';')
               break;
            digit[y++]=ch;
            m++;
            ch=buffer[m];
         }
         digit[y]='\0';
         fprintf(output,"%7d\t\t
%7d\tDigit\t\t    %s\n",l,t,digit);
         t++;
         bzero(digit,100);
      }
      else if(isalpha(ch)){
         int y=0;
         while(isalpha(ch)){
            if(ch==' ' || ch==';' || ch==',')
               break;
            identifier[y++]=ch;
            m++;
            ch=buffer[m];
         }
         if(keyword_checker(identifier)==1){
            fprintf(output,"%7d\t\t
%7d\tKeyword\t\t    %s\n",l,t,identifier);
         }
         else{
            fprintf(output,"%7d\t\t
%7d\tIdentifier\t\t    %s\n",l,t,identifier);
         }
         t++;
         bzero(identifier,100);
      }
   m++;
   }
return 0;
}
int operator_checker(char op){
   if( op=='+' || op== '-' || op=='*' || op=='/'){
      return 1;
   }
   Else{
      return 0;
   }
}
int special_checker(char sp){
   if( sp==';' || sp=='{' || sp=='}' || sp=='(' ||
sp==')' || sp=='?' || sp=='@' || sp=='!' ||
sp=='%')
   {
      return 1;
```

```
    }
    else{
        return 0;
    }
}
int keyword_checker(char keyword[])
{
    char keywords[32][10] =
{"auto","break","case","char","const","contin
ue","default","do","double","else","enum","e
xtern","float","for","goto","if","int","long","re
gister","return","short","signed","sizeof","stat
ic","struct","switch","typedef","union","unsig
ned","void","volatile","while"};
    int i, flag = 0;
    for(i = 0; i < 32;i++){
        if(strcmp(keywords[i], keyword) == 0){
            flag = 1;
            break;
        }
    }
return flag;
}
```

Output

```
Input file contents are :
void main()
{
    int sum a,b;
    sum = a + b;
    printf("%d",sum);
}
```

Input.txt

input.txt

```
void main()
{
    int sum a,b;
    sum = a + b;
    printf("%d",sum);
}
```

Output.txt

output.txt

Line no.	Token no.	Token	Lexeme
1	1	Keyword	void
1	2	Identifier	main
1	3	Special symbol	(
1	4	Special symbol	)
2	5	Special symbol	{
3	6	Keyword	int
3	7	Identifier	sum
3	8	Identifier	a
3	9	Identifier	b
4	10	Identifier	sum
4	11	Identifier	a
4	12	Operator	+
4	13	Identifier	b
5	14	Identifier	printf
5	15	Special symbol	%
5	16	Identifier	d
5	17	Identifier	sum
6	18	Special symbol	}

Program – Shift Reduce Parser

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char g1[]="E+E";
char g2[]="E*E";
char g3[]="i";
int handle_checker(char stack[],int len);
int main(){
   char expression[30];
   char stack[500],ch;
   char buffer[2];
   buffer[1]='\0';
   int m=0,retval;
   printf("Enter the arithmetic expression : ");
   scanf("%s",expression);
   int len=strlen(expression);
   printf("Stack \t Input \t   Action\n");
   int i=0;
   while(i!=len+1){
      ch=expression[i];
      buffer[0]=ch;
      printf("\n%s",stack);
      printf("\t");
      for(int o=i;o<len;o++){
         printf("%c",expression[o]);}
      printf("\t");
      retval=handle_checker(stack,len);
      if(retval==0){
         strcat(stack,buffer);
         i++;
         printf("  Shift\n");}
      else{
         printf("  Reduce\n");}}
   if(strcmp(stack,"E")==0 && i>=len){
      printf("\nAccepted");}
   else{
      printf("\nRejected");}
return 0;}
int handle_checker(char stack[],int len){
   int i=len-1,m=0,y=1,d;
   char buffer[100];
   while(y==1){
      d=i;m=0;
      while(i!=len){
         buffer[m++]=stack[i];
         i++;}
      if(strcmp(buffer,g1)==0 ||
strcmp(buffer,g2)==0 ||
strcmp(buffer,g3)==0){
         i=d;
         while(i!=len){
            stack[i]='\0';
            i++;}
         strcat(stack,"E");
         return 1;}
      bzero(buffer,len);
      i=d;
      i--;
      if(i==-1){
         break;}
   return 0;
}
```

Output

```
Enter the arithmetic expression : i+i*i
Stack    Input     Action

         i+i*i     Shift

i        +i*i      Reduce

E        +i*i      Shift

E+       i*i       Shift

E+i      *i        Reduce

E+E      *i        Reduce

E        *i        Shift

E*       i         Shift

E*i                Reduce

E*E                Reduce

E                  Shift

Accepted
Enter the arithmetic expression : i+*i
Stack    Input     Action

         i+*i      Shift

i        +*i       Reduce

E        +*i       Shift

E+       *i        Shift

E+*      i         Shift

E+*i               Shift

Rejected
```

Program – Intermediate Code Generation

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#include<stdio.h>
#include<string.h>
char stack[100];
int top=-1;
void push(char ch){
   if(top==100){
      printf("\nStack is full");
   }
   else{
      top++;
      stack[top]=ch;
   }
}
char pop(){
   char item;
   if(top==-1){
      printf("\nStack is empty");
   }
   else{
      item=stack[top];
      top--;
   }
   return item;
}
int isOperator(char ch){
   if(ch=='+' || ch=='-' || ch=='/' || ch=='*' ||
ch=='^'){
      return 1;
   }
   else
      return 0;
}
int precedence(char ch){
   if(ch=='^')
      return 3;
   else if(ch=='/' || ch=='*')
      return 2;
   else if(ch=='+' || ch=='-')
      return 1;
   return 0;
}
int isOperand(char ch){
   if((ch>=97 && ch<=122) || (ch>=65 &&
ch<=90)){
      return 1;
   }
   else
      return 0;
}
int main(){
   char exp[50],ch;
   char output[50];
   int i,j,len,k=0;
   printf("\nEnter the expression : ");
   scanf("%s",exp);
   len=strlen(exp);
   for(i=0;i<len;i++){
      ch=exp[i];
      if(isOperand(ch)){
         output[k++]=ch;
      }
      else if(isOperator(ch)){
         if(top==-1){
            push(ch);
         }
         else{
            char temp;
            int flag=0;
            while(top>-1 &&
isOperator(stack[top]) &&
precedence(stack[top])>=precedence(ch)){
               temp=pop();
               output[k++]=temp;
            }
            push(ch);
         }
      }
   }
   while(top!=-1){
      char temp=pop();
      output[k++]=temp;
   }
   printf("\nPostfix is : %s",output);
   printf("\n\nIntermediate code is : \n");
   char a,b,op,a1='\0',b1='\0';
   int c=1;
   for(i=0;i<len;i++){
      ch=output[i];
      if(isOperand(ch))
      {
         push(ch);
      }
      else if(isOperator(ch)){
         a=pop();
         if(a=='t')
            a1=pop();
```

```
        b=pop();
        if(b=='t')
           b1=pop();

        printf("\nt%d = %c%c %c
%c%c",c,b,b1,ch,a,a1);
        char m=c+'0';
        push(m);
        push('t');
        c++;
      }
   }
   printf("\n");
   return 0;
}
```

Output

```
Enter the expression : a+b*c

Postfix is : abc*+

Intermediate code is :

t1 = b * c
t2 = a + t1
```

Program – NFA To DFA Conversion

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int outRow=0,outCol=0;
int states,variables;
char state[10][10];
char variable[10][10];
char outputTable[20][10][1000]={'\0'};
char inputTable[10][10][1000];
char outputStates[20][200];
int it=0;
int outputStatesChecker(char state[]){
   for(int i=0;i<it;i++){
      if(strcmp(state,outputStates[i])==0){
         return 1;
      }
   }
   return 0;
}
int  compositeChecker(char state[]){
   int len=strlen(state);
   if(len>=4){
      return 1;
   }
   return 0;
}
void splitterAndAdder(char state[]){
   int splitar[20],k=0;
   for(int i=1;i<strlen(state);i+=2){
      splitar[k]=state[i]-'0';k++;
   }
   for(int i=0;i<variables;i++){
      char tempAr[100];
      bzero(tempAr,100);
      for(int j=0;j<k;j++){
         if(strcmp(inputTable[splitar[j]-
1][i],"*")==0){
            continue;
         }
         strcat(tempAr,inputTable[splitar[j]-
1][i]);
      }
      strcpy(outputTable[outRow][i],tempAr);
   }
   outRow++;
}
void displayOutput(){
   int m=0,k=0;
   printf("\n\nOutput Table\n\n");
   printf("\t");
   for(int i=0;i<variables;i++){
      printf("%c    ",'a'+k);
      k++;
   }
   printf("\n");
   for(int i=0;i<outRow;i++){
      printf("\n%s\t",outputStates[m]);
      m++;
      for(int j=0;j<variables;j++){
         printf("%s  ",outputTable[i][j]);
      }
      printf("\n");
   }
}
void displayInput(){
   int m=1,k=0;
   printf("\n\nInput Table\n\n");
   printf("\t");
   for(int i=0;i<variables;i++){
      printf("%c    ",'a'+k);
      k++;
   }
   printf("\n");
   for(int i=0;i<states;i++){
      printf("\n%s%d\t","q",m);
      m++;
      for(int j=0;j<variables;j++){
         if( (strcmp(inputTable[i][j],"*"))==0  ){
            printf("%s    ",inputTable[i][j]);
         }
         else{
            printf("%s  ",inputTable[i][j]);
         }
      }
      printf("\n");
   }
}
void main(){
   printf("\nEnter the number of states and
states : ");
   scanf("%d",&states);
```

```
    for(int i=0;i<states;i++)
    scanf("%s",state[i]);
    printf("Enter the number of variables and
variables : ");
    scanf("%d",&variables);
    for(int i=0;i<variables;i++)
    scanf("%s",variable[i]);
    printf("Enter the Table values\n");
    for(int i=0;i<states;i++){
        for(int j=0;j<variables;j++){
            scanf("%s",inputTable[i][j]);
        }
    }
    displayInput();
    for(int j=0;j<variables;j++)
    strcpy(outputTable[0][j],inputTable[0][j]);
    outRow++;
    strcpy(outputStates[it],"q1");
    it++;
    for(int row=0;row<states*2;row++){
        for(int col=0;col<variables;col++){
            if(outputTable[row][col][0]=='\0'){
                displayOutput();
                exit(0);
            }
            char tempStr[50];
            strcpy(tempStr,outputTable[row][col]);
            int val = outputStatesChecker(tempStr);
            if(val==0){
                strcpy(outputStates[it],tempStr);
                it++;
                int retval =
compositeChecker(tempStr);
                if(retval==0){
                    int index=tempStr[1]-'0';
                    for(int m=0;m<variables;m++){
                        strcpy(outputTable[outRow][m],i
nputTable[index][m]);
                    }
                    outRow++;
                }
                else{
                    splitterAndAdder(tempStr);
                }
            }
        }
    }
}
```

Output

```
Enter the number of states and states : 3
q1 q2 q3
Enter the number of variables and variables : 2
a b
Enter the Table values
q1 q1q2
*  q3
*  *


Input Table

        a    b

q1      q1   q1q2

q2      *    q3

q3      *    *


Output Table

        a    b

q1      q1   q1q2

q1q2    q1   q1q2q3

q1q2q3  q1   q1q2q3
```

Program – Constant Propagation

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#include<stdio.h>
#include<ctype.h>
int n;
char prod[10][20];
void input()
{
   int i;
   printf("Enter number of productions : ");
   scanf("%d",&n);
   printf("\nEnter %d productions : \n",n);
   for(i=0;i<n;i++)
   {
      scanf("%s",prod[i]);
   }
}
void replace(char c ,char num, int no)
{
   int i;
   for(i=no+1;i<n;i++)
   {
      int j=2;
      if(prod[i][0]==c)
         break;
      while(prod[i][j]!='\0')
      {
         if(prod[i][j]==c)
         {
            prod[i][j]=num;
            break;
         }
      j++;
      }
   }
}
void check()
{
   int i;
   for(i=0;i<n;i++)
   {
      if(isdigit(prod[i][2]) && prod[i][3]=='\0')
      {
         replace(prod[i][0],prod[i][2],i);
         prod[i][0]='#';
      }
   }
}
void display()
{
   for(int i=0;i<n;i++)
   {
      if(prod[i][0]!='#')
      {
         printf("%s\n",prod[i]);
      }
   }
}
void main()
{
   input();
   check();
   printf("\nOptimised code is : \n");
   display();
}
```

Output

```
Enter number of productions : 4

Enter 4 productions :
a=3
c=a+b
b=6
d=a+b

Optimised code is :
c=3+b
d=3+6
```

Program – Target Code Generation

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

#include<stdio.h>
#include<string.h>
int main()
{
   char exps[20][20],i=0,j=0,op[5];
   int count=1;
   printf("\nEnter the Intermediate codes (exit
to stop)\n");
   while(j==0)
   {
      char exp[20];
      scanf("%s",exp);
      if((strcmp(exp,"exit"))==0)
      {
         break;
      }
      strcpy(exps[i++],exp);
   }
   printf("\nThe Assembly code is\n");
   while(j!=i)
   {
      switch(exps[j][3])
      {
         case '+':
         {
            strcpy(op,"ADD");
            break;
         }
         case '-':
         {
            strcpy(op,"SUB");
            break;
         }
         case '/':
         {
            strcpy(op,"DIV");
            break;
         }
         case '*':
         {
            strcpy(op,"MUL");
            break;
         }
      }
      printf("\n\tMOV
R%d,%c",count,exps[j][2]);
      printf("\n\t%s
R%d,%c",op,count,exps[j][4]);
      printf("\n\tMOV
%c,R%d",exps[j][0],count);
      j++;
      count++;
   }
   printf("\n");
   return 0;
}
```

Output

```
Enter the Intermediate codes (exit to stop)
a=b+c
c=d+e
exit

The Assembly code is

        MOV R1,b
        ADD R1,c
        MOV a,R1
        MOV R2,d
        ADD R2,e
        MOV c,R2
```

Program – Lexical Analyzer

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

%{
#include<stdio.h>
#include<string.h>
char
key[100][100],head[100][100],dig[100][100],o
p[100][100],id[100][100],lit[100][100];
int
i=0,j=0,k=0,l=0,a=0,b=0,c=0,d=0,m=0,n=0,sz=0
,count=0;
%}
KW
"int"|"while"|"if"|"else"|"for"|"char"|"float"
|"case"|"switch"|"printf"|"scanf"|"void"
HF "#include<".*">"
OP "+"|"-"|"*"|"/"|"="
DIG [0-9]*|[0-9]*"."[0-9]+
ID [a-zA-Z][a-zA-Z0-9]*
LI "\"".*"\""
%%
{KW} {strcpy(key[i],yytext);i++;}
{HF} {strcpy(head[j],yytext);j++;}
{DIG} {strcpy(dig[k],yytext);k++;}
{OP} {strcpy(op[sz],yytext);sz++;}
{ID} {strcpy(id[n],yytext);n++;}
{LI} {strcpy(lit[count],yytext);n++;}
. {}
%%
int main()
{
   yyin=fopen("input.txt","r+");
   yylex();
   printf("\nThe keywords :\n");
   for(b=0;b<i;b++)
   {
      for(m=b+1;m<i;m++)
      {
         if(strcmp(key[b],key[m])==0)
         {
            for(c=m;c<i-1;c++)
            {
               strcpy(key[c],key[c+1]);
            }
            i--;
            m--;
         }
      }
   }
}
for(a=0;a<i;a++)
{
   printf("%s\n",key[a]);
}
printf("\nThe headerfile :\n");
for (a=0;a<j;a++)
{
   printf("%s\n",head[a]);
}

printf("\nThe digits :\n");
for(a=0;a<k;a++)
{
   printf("%s\n",dig[a]);
}
printf("\nOperators :\n");
for(b=0;b<sz;b++)
{
   for(m=b+1;m<sz;m++)
   {
      if(strcmp(op[b],op[m])==0)
      {
         for(c=m;c<sz-1;c++)
         {
            strcpy(op[c],op[c+1]);
         }
         sz--;
         m--;
      }
   }
}
for (a=0;a<sz;a++)
{
   printf("%s\n",op[a]);
}
printf("\nIdentifiers :\n");
for(b=0;b<n;b++)
{
   for(m=b+1;m<n;m++)
   {
      if(strcmp(id[b],id[m])==0)
      {
         for(c=m;c<n-1;c++)
         {
```

```
            strcpy(id[c],id[c+1]);
        }
        n--;
        m--;
      }
    }
  }
  for(a=0;a<n;a++)
  {
    printf("%s\n",id[a]);
  }
}
int yywrap()
{
  return 1;
}
```

Input.txt

```
#include<stdio.h>
void main()
{
      int a,b,sum;
      a=10;
      b=20;
      sum=a+b;
      printf("Sum is : %d",sum);
}
```

Output

```
The keywords :
void
int
printf

The headerfile :
#include<stdio.h>

The digits :
10
20

Operators :
=
+

Identifiers :
main
a
b
sum
```

Program – Arithmetic Exp Validation

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */
```

Lex Program

```
%{
#include"y.tab.h"
extern yylval;
%}
%%
[0-9]+ {yylval=atoi(yytext); return NUMBER;}
[a-zA-Z]+ {return ID;}
[\t]+ ;
\n {return 0;}
. {return yytext[0];}
%%
```

Yacc Program

```
%{
#include<stdio.h>
%}
%token NUMBER ID
%left '+' '-'
%left '*' '/'
%%
expr: expr '+' expr | expr '-' expr |expr '*' expr
|expr '/' expr |'-'NUMBER |'-'ID |'('expr')' |
NUMBER | ID ;
%%
main()
{
        printf("Enter the expression : ");
        yyparse();
        printf("\nExpression is valid\n");
        exit(0);
}
int yyerror(char *s)
{
        printf("\nExpression is invalid\n");
        exit(0);
}
```

Output

```
Enter the expression : a*c+b
Expression is valid
```

```
Enter the expression :  a++b
Expression is invalid
```

Program – Character, Word and Line Count

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

%{
#include<stdio.h>
int sc=0,wc=0,lc=0,cc=0;
%}
nl [\n]
sp [ \t]
word [^\t\n ]+
%%
{nl} { lc++;}
{sp} { sc++; cc+=yyleng;}
{word} { wc++;  cc+=yyleng;}
%%
int main(int argc ,char* argv[ ])
{
        printf("Enter the input : \n");
        yylex();
        printf("The number of lines = %d\n",lc);
        printf("The number of words = %d\n",wc);
        printf("The number of characters are = %d\n",cc);
}
int yywrap( )
{
        return 1;
}
```

Output

```
Enter the input :
Hello
How are you
The number of lines = 2
The number of words = 4
The number of characters are = 16
```

Program – Vowel & Consonant Count

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */

%{
        #include<stdio.h>
        int vow_count=0;
        int const_count =0;
%}
vow [aeiouAEIOU]
const [a-zA-Z]
%%
{vow} {vow_count++;}
{const} {const_count++;}
%%
int main()
{
   printf("Enter the string : ");
   yylex();
   printf("Number of vowels :  %d\n", vow_count);
   printf("Number of consonants :  %d\n", const_count);
   return 0;
}
int yywrap()
{
        return 1;
}
```

Output

```
Enter the string : Compiler design lab

Number of vowels :  6
Number of consonants :  11
```

Program – Calculator

```
/* Ashiq Cherian
   CSE – A, S7
   Roll No : 36 */
```

Lex Program

```
%{
   #include<stdio.h>
   #include "y.tab.h"
   extern int yylval;
%}
%%
[0-9]+ { yylval=atoi(yytext); return NUMBER;}
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
   return 1;
}
```

Yacc Program

```
%{
   #include<stdio.h>
   int flag=0;
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
printf("\nResult=%d\n",$$); return 0;};
E:E'+'E {$$=$1+$3;}|E'-'E {$$=$1-$3;}|E'*'E
{$$=$1*$3;}|E'/'E {$$=$1/$3;}|E'%'E
{$$=$1%$3;}|'('E')' {$$=$2;}| NUMBER
{$$=$1;};
%%
void main()
{
  printf("\nEnter an Arithmetic Expression : ");
  yyparse();
}
void yyerror()
{
  printf("\nEntered arithmetic expression is
Invalid\n");
  flag=1;
}
```

Output

```
Enter an Arithmetic Expression : 3*2+6

Result=12
```

```
Enter an Arithmetic Expression : 6+3*/5

Entered arithmetic expression is Invalid
```