PROGRAM 1(a):

Write a LEX program to recognize valid arithmetic expression. Identifiers in the expression could be only integers and operators could be + and *. Count the identifiers and operators present and print them separately.

```
%{
#include<stdio.h>
intnid=0,np=0,nm=0,nmul=0,nd=0;
int flag1=0,flag2=0;
%}
%%
[[] {flag2++;}
[]] {flag2--;}
[a-zA-Z 0-9]+ {nid++;flag1++;}
[+] {np++;flag1--;}
[-] {nm++;flag1--;}
[*] {nmul++;flag1--;}
[/] {nd++;flag1--;}
%%
main()
{
printf("Enter the arithematic operation");
yylex();
if(flag2!=0 || flag1!=1)
{
printf("Invalid expression \n");
return(0);
}
else
```

```
{
printf("valid arithematic expression \n");
printf("no. of identifiers are %d \n",nid);
printf("no. of plus symbols are %d \n",np);
printf("no. of minus symbols are %d \n",nm);
printf("no. of multiplication symbols are %d \n",nmul);
printf("no. of divison symbols are %d \n",nd);
}
}
```

Output:

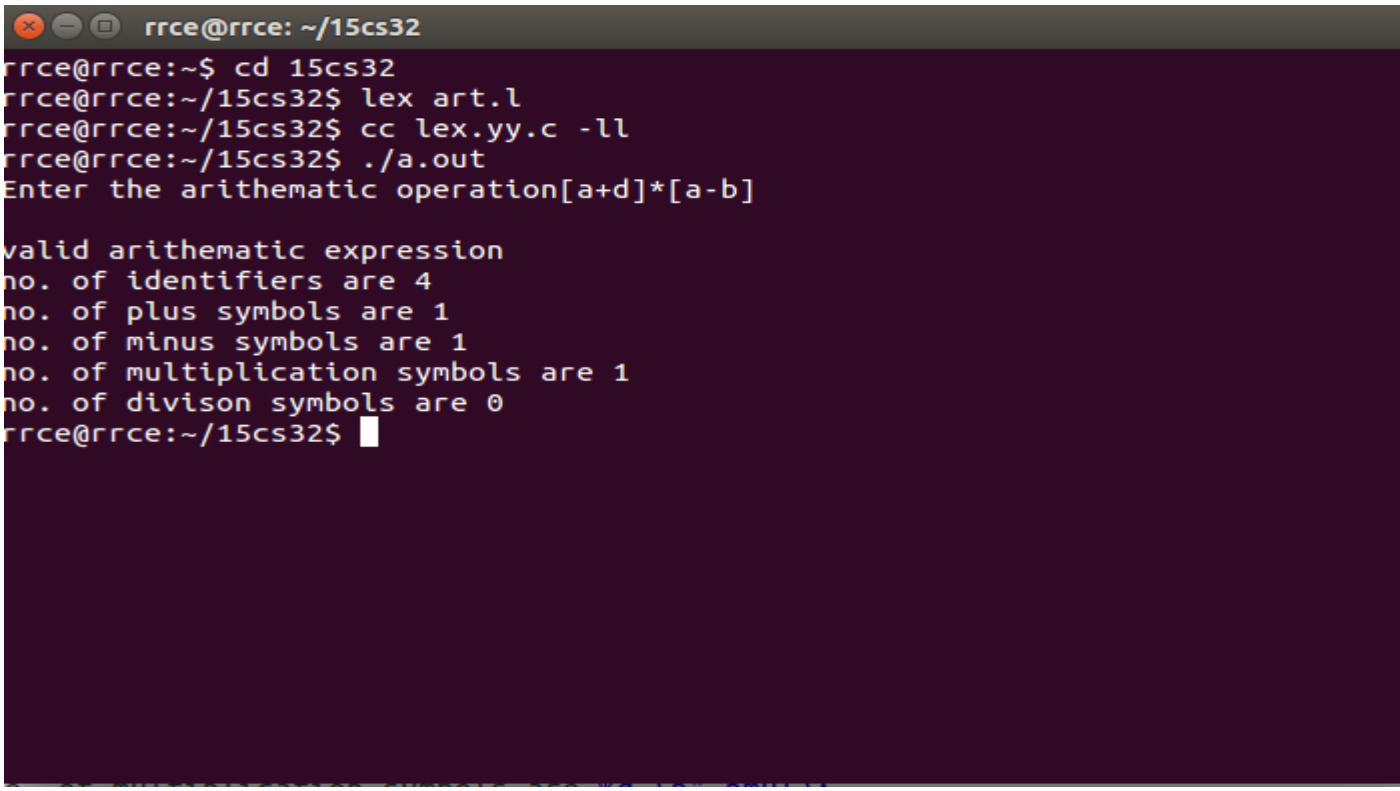
cd (directory name)

lex (file name) .l

cc lex.yy.c –ll

./a.out

Press cntrl d to get the ouput ; Take an output for invalid expression also.



PROGRAM 1(b):

Write YACC program to evaluate arithmetic expression involving operators: +, -, *, and /

LEX PART:

```
%{
#include "y.tab.h"
externintyylval;
%}
%%
[0-9]*  {yylval=atoi(yytext);
return NUM;
        }
[\t]    ;
\n      return 0;
.       returnyytext[0];
%%
```

YACC PART:

```
%{
#include<stdio.h>
%}
%token NUM
%left '+' '-'
%left '*' '/'
%left '(' ')'
%token .
%%

expr:e {printf("\n Result:%d\n",$$);
return 0;
```

```
    }
e:e'+'e  {$$=$1+$3;}
|e'-'e   {$$=$1-$3;}
|e'*'e   {$$=$1*$3;}
|e'/'e   {$$=$1/$3;}
|'('e')' {$$=$2;}
|NUM     {$$=$1;}
|.
%%

main()
{
printf("enter the arithmetic expression \n");
yyparse();
printf("\n valid expression \n");
}
yyerror()
{
printf("\n Invalid expression \n");
exit(0);
}
```

OUTPUT:

lex (file name).l

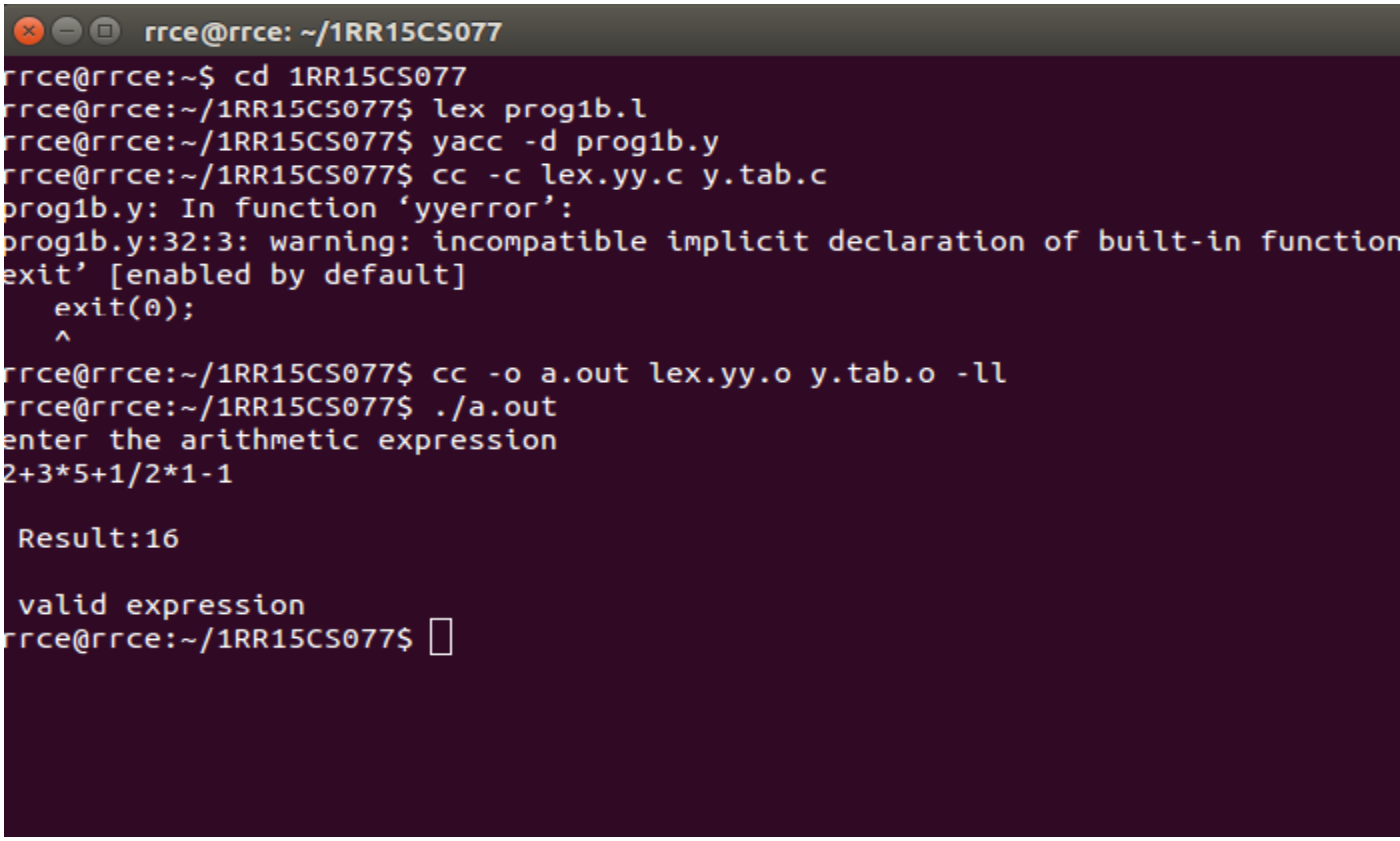
yacc –d (file name).y

cc –c lex.yy.cy.tab.c

cc –o a.outlex.yy.oy.tab.o -ll

./a.out

Take an output for invalid expression also.


rrce@rrce: ~/1RR15CS077
rrce@rrce:~$ cd 1RR15CS077
rrce@rrce:~/1RR15CS077$ lex prog1b.l
rrce@rrce:~/1RR15CS077$ yacc -d prog1b.y
rrce@rrce:~/1RR15CS077$ cc -c lex.yy.c y.tab.c
prog1b.y: In function ‘yyerror’:
prog1b.y:32:3: warning: incompatible implicit declaration of built-in function
exit’ [enabled by default]
exit(0);
^
rrce@rrce:~/1RR15CS077$ cc -o a.out lex.yy.o y.tab.o -ll
rrce@rrce:~/1RR15CS077$ ./a.out
enter the arithmetic expression
2+3*5+1/2*1-1
Result:16
valid expression
rrce@rrce:~/1RR15CS077$

PROGRAM 2:

Develop, implement and execute a program using YACC tool to recognize all strings ending with b preceded by n a's using the grammar a^nb(a power n) (note:input n value)

LEX PART:

```
%{
#include "y.tab.h"
%}
%%
[a]    return A;
[b]    return B;
[\n] return yytext [0];
%%
```

YACC PART:

```
%{
#include <stdio.h>
%}
%token A B
%%
str:s '\n'
s: A s1 B|B
s1: ; | A s1
%%
int main()
{
printf("enter the string \n");
if(yyparse()==0)
printf("\n valid string\n");
return 0;
```

```
}
yyerror()
{
printf("\n invalid string\n");
return 1;
}
```

OUTPUT:

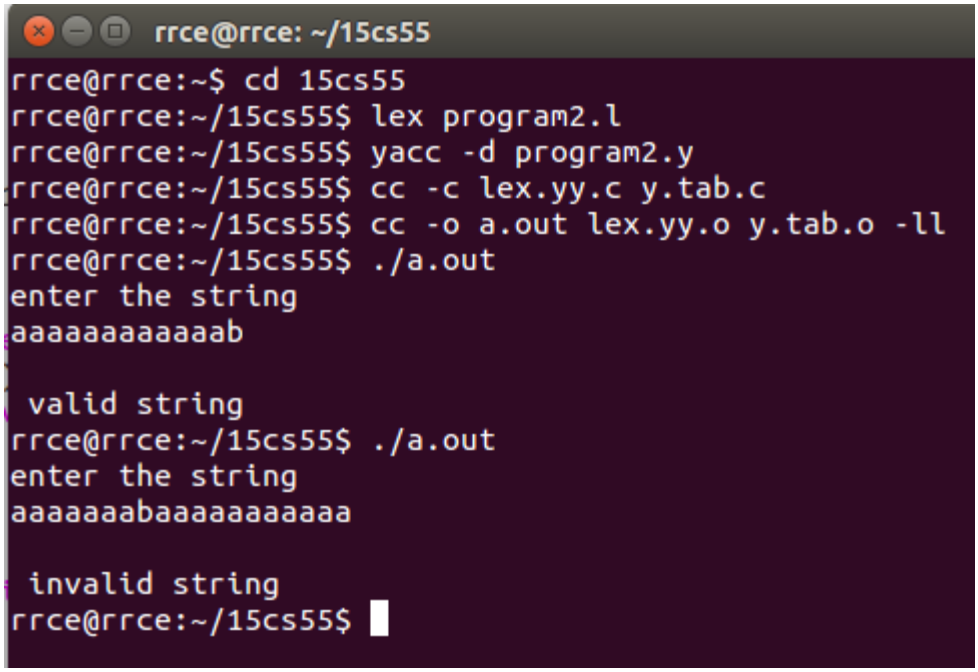
lex (file name).l

yacc –d (file name).y

cc -c lex.yy.cy.tab.c

cc -o a.outlex.yy.oy.tab.o -ll

./a.out



PROGRAM3:

Design, develop and implement YACC/C program to construct Predictive / LL(1) Parsing Table for the grammar rules: A → aBa , B → bB | ε . Use this table to parse the sentence: abba$

```
#include<stdio.h>
#include<string.h>
void main()
{
char fin[10][20],st[10][20],ft[20][20],fol[20][20];
int a=0,e,i,t,b,c,n,k,l=0,j,s,m,p;
printf("enter the no. of coordinates \n");
scanf("%d",&n);
printf("enter the production in a grammer\n");
for(i=0;i<n;i++)
scanf("%s",st[i]);
for(i=0;i<n;i++)
fol[i][0]='\0';
for(s=0;s<n;s++)
{
for(i=0;i<n;i++)
{
j=3;
l=0;
a=0;
l1:if(!((st[i][j]>64)&&(st[i][j]<91)))
{
for(m=0;m<l;m++)
{
if(ft[i][m]==st[i][j])
```

```
goto s1;
}
ft[i][l]=st[i][j];
l=l+1;
s1:j=j+1;
}
else
{
if(s>0)
{
while(st[i][j]!=st[a][0])
{
a++;
}
b=0;
while(ft[a][b]!='\0')
{
for(m=0;m<l;m++)
{
if(ft[i][m]==ft[a][b])
goto s2;
}
ft[i][l]=ft[a][b];
l=l+1;
s2:b=b+1;
}
}
}
while(st[i][j]!='\0')
```

```
{
if(st[i][j]=='|')
{
j=j+1;
goto l1;
}
j=j+1;
}
ft[i][l]='\0';
}
}
printf("first pos\n");
for(i=0;i<n;i++)
printf("FIRS[%c]=%s\n",st[i][0],ft[i]);
fol[0][0]='$';
for(i=0;i<n;i++)
{
k=0;
j=3;
if(i==0)
l=1;
else
l=0;

k1:while((st[i][0]!=st[k][j])&&(k<n))
{
if(st[k][j]=='\0')
{
k++;
```

```
j=2;
}
j++;
}
j=j+1;
if(st[i][0]==st[k][j-1])
{
if((st[k][j]!='|')&&(st[k][j]!='\0'))
{
a=0;
if(!((st[k][j]>64)&&(st[k][j]<91)))
{
for(m=0;m<l;m++)
{
if(fol[i][m]==st[k][j])
goto q3;
}
q3:
fol[i][l]=st[k][j];
l++;
}
else
{
while(st[k][j]!=st[a][0])
{
a++;
}
p=0;
while(ft[a][p]!='\0')
```

```
{
if(ft[a][p]!='@')
{
for(m=0;m<l;m++)
{
if(fol[i][m]==ft[a][p])
goto q2;
}
fol[i][l]=ft[a][p];
l=l+1;
}
 else
e=1;
q2:p++;
}
if(e==1)
{
e=0;
goto a1;
}
}
}
else
{
a1:c=0;
a=0;
while(st[k][0]!=st[a][0])
{
a++;
```

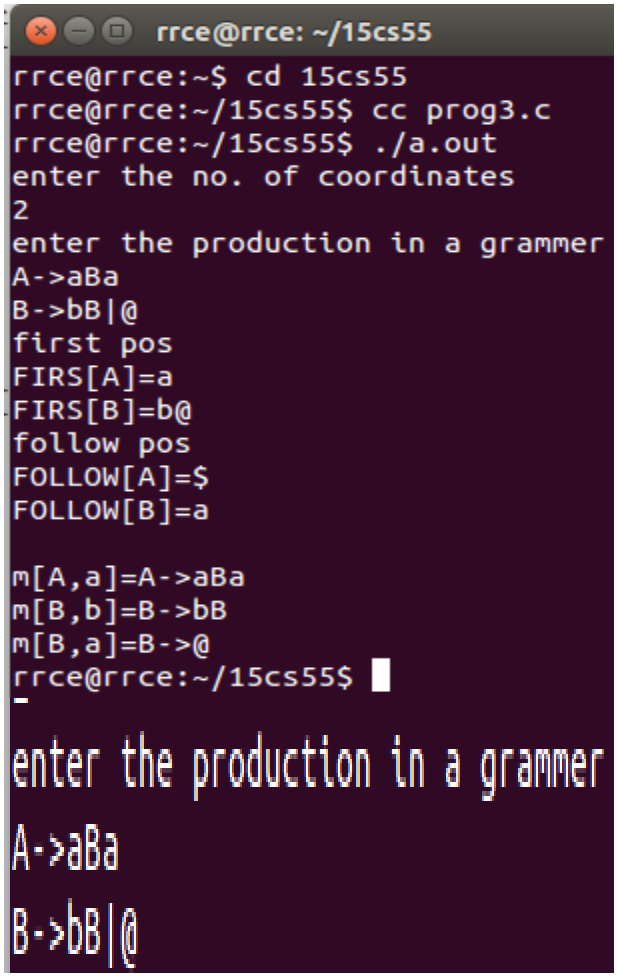
```
}
while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
{
for(m=0;m<l;m++)
{
if(fol[i][m]==fol[a][c])
goto q1;
}
fol[i][j]=fol[a][c];
l++;
q1:c++;
}
}
goto k1;
}
fol[i][l]='\0';
}
printf("follow pos\n");
for(i=0;i<n;i++)
printf("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
printf("\n");
s=0;
for(i=0;i<n;i++)
{
j=3;
while(st[i][j]!='\0')
{
if((st[i][j-1]=='|')||(j==3))
{
```

```
for(p=0;p<=2;p++)
{
fin[s][p]=st[i][p];
}
t=j;
for(p=3;((st[i][j]!='|')&&(st[i][j]!='\0'));p++)
{
fin[s][p]=st[i][j];
j++;
}
fin[s][p]='\0';
if(st[i][t]=='@')
{
b=0;
a=0;
while(st[a][0]!=st[i][0])
{
a++;
}
while(fol[a][b]!='\0')
{
printf("m[%c,%c]=%s\n",st[i][0],fol[a][b],fin[s]);
b++;
}
}
else if(!((st[i][t]>64)&&(st[i][t]<91)))
printf("m[%c,%c]=%s\n",st[i][0],st[i][t],fin[s]);
else
{
```

```
b=0;
a=0;
while(st[a][0]!=st[i][3])
{
a++;
}
while(ft[a][b]!='\0')
{
printf("m[%c,%c]=%s\n",st[i][0],ft[a][b],fin[s]);
b++;
}
}
s++;
}
if(st[i][j]=='|')
j++;
}
}
}
```

OUTPUT: p

```
cd (directory name)
cc (filename).c
./a.out
```


rrce@rrce: ~/15cs55
rrce@rrce:~$ cd 15cs55
rrce@rrce:~/15cs55$ cc prog3.c
rrce@rrce:~/15cs55$ ./a.out
enter the no. of coordinates
2
enter the production in a grammer
A->aBa
B->bB|@
first pos
FIRS[A]=a
FIRS[B]=b@
follow pos
FOLLOW[A]=$
FOLLOW[B]=a
m[A,a]=A->aBa
m[B,b]=B->bB
m[B,a]=B->@
rrce@rrce:~/15cs55$
enter the production in a grammer
A->aBa
B->bB|@

PROGRAM 4:

Design, develop and implement YACC/C program to demonstrate Shift Reduce Parsing technique for the grammar rules: E → E+T | T, T → T*F | F, F → (E) | id and parse the sentence: id + id * id.

```
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
void main()
{
puts("GRAMMAR Is E->E+E\nE->E*E\nE->(E)\nE->id");
puts("enter the input string");
gets(a);
c=strlen(a);
strcpy(act,"SHIFT->");
puts("stack \tinput \t\taction");
for(k=0,i=0;j<c;k++,i++,j++)
{
if(a[j]=='i'&&a[j+1]=='d')
{
stk[i]=a[j];
stk[i+1]=a[j+1];
stk[i+2]='\0';
a[j]=' ';
a[j+1]=' ';
printf("\n$%s\t%s$\t%sid",stk,a,act);
check();
}
```

```
else
{
stk[i]=a[j];
stk[i+1]='\0';
a[j]=' ';
printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
check();
}
}
}
void check()
{
strcpy(ac,"REDUCE TO E");
for(z=0;z<c;z++)
if(stk[z]=='i'&&stk[z+1]=='d')
{
stk[z]='E';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
j++;
}
for(z=0;z<c;z++)
if(stk[z]=='E'&&stk[z+1]=='+'&&stk[z+2]=='E')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+2]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
i=i-2;
```

```
}
for(z=0;z<c;z++)
if(stk[z]=='E'&&stk[z+1]=='*'&&stk[z+2]=='E')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
i=i-2;
}
for(z=0;z<c;z++)
if(stk[z]=='('&&stk[z+1]=='E'&&stk[z+2]==')')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
i=i-2;
}
}
```

OUTPUT:

cd (directory name)
cc (filename).c
./a.out

```
rrce@rrce:~$ cd 15cs55
rrce@rrce:~/15cs55$ cc prog4.c
prog4.c: In function ‘main’:
prog4.c:10:1: warning: ‘gets’ is deprecated (declared at /usr/include/stdio.h
8) [-Wdeprecated-declarations]
 gets(a);
 ^
/tmp/cc05cCVw.o: In function `main':
prog4.c:(.text+0x29): warning: the `gets' function is dangerous and should no
e used.
rrce@rrce:~/15cs55$ ./a.out
GRAMMAR Is E->E+E
E->E*E
E->(E)
E->id
enter the input string
id+id*id
stack   input           action

$id       +id*id$       SHIFT->id
$E        +id*id$       REDUCE TO E
$E+        id*id$       SHIFT->symbols
$E+id        *id$       SHIFT->id
$E+E         *id$       REDUCE TO E
$E           *id$       REDUCE TO E
$E*           id$       SHIFT->symbols
$E*id           $       SHIFT->id
$E*E            $       REDUCE TO E
$E              $       REDUCE TO Errce@rrce:~/15cs55$
```

PROGRAM 5:

Design, develop and implement a C/Java program to generate the machine code using triples for the statement

A= -B*(C+D) whose intermediate code in three-address form:

T1= -B

T2= C+D

T3= T1*T2

A= T3

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
char op[2],arg1[5],arg2[5],res[5];
void main()
{
FILE *fp1,*fp2;
fp1=fopen("input.txt","r");
fp2=fopen("output.txt","w");
while(!feof(fp1))
{
fscanf(fp1,"%s%s%s%s",res,arg1,op,arg2);
if(strcmp(op,"+")==0)
{
fprintf(fp2,"\n MOV R0,%s",arg1);
fprintf(fp2,"\n ADD R0,%s",arg2);
fprintf(fp2,"\n MOV %s,R0",res);
}
if(strcmp(op,"*")==0)
{
fprintf(fp2,"\n MOV R0,%s",arg1);
```

```
fprintf(fp2,"\n MUL R0,%s",arg2);
fprintf(fp2,"\n MOV %s,R0",res);
}
if(strcmp(op,"-")==0)
{
fprintf(fp2,"\n MOV R0,%s",arg1);
fprintf(fp2,"\n SUB R0,%s",arg2);
fprintf(fp2,"\n MOV %s,R0",res);
}
if(strcmp(op,"/")==0)
{
fprintf(fp2,"\n MOV R0,%s",arg1);
fprintf(fp2,"\n DIV R0,%s",arg2);
fprintf(fp2,"\n MOV %s,R0",res);
}
if(strcmp(op,"=")==0)
{
fprintf(fp2,"\n MOV R0,%s",arg1);
//fprintf(fp2,"\n ADD R0,%s",arg2);
fprintf(fp2,"\n MOV %s,R0",res);
}
}
fclose(fp1);
fclose(fp2);
//getch();
}
```

Save the program in **.c** extension.

CREATE AN INPUT FILE (input.txt).

input.txt

```
T1 -B = ?
T2 C + D
T3 T1 * T2
A T3 =
```

CREATE AN OUTPUT FILE(output.txt).

OUTPUT:

cc (file name).c

./a.out input.txt output.txt

gedit output.txt

```
rrce@rrce: ~/50
rrce@rrce:~$ cd 50
rrce@rrce:~/50$ cc pgrm3.c
rrce@rrce:~/50$ ./a.out input.txt output.txt
rrce@rrce:~/50$ gedit output.txt
rrce@rrce:~/50$
```

```
MOV R0,-B
MOV T1,R0
MOV R0,C
ADD R0,D
MOV T2,R0
MOV R0,T1
MUL R0,T2
MOV T3,R0
MOV R0,T3
MOV A,R0
```

PROGRAM 6(A):

Write a LEX program to eliminate comment lines in a C program and copy the resulting program into a separate file.

```
%{
#include<stdio.h>
int c=0;
%}
%%
[/][*][^."*/"]*[*][/]   {c++;fprintf(yyout," ");}
"//".*                  {c++;fprintf(yyout," ");}
%%
int main(intargc,char *argv[])
{
if(argc!=3)
{
printf("invalid input");
return 0;
}
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");
yylex();
printf("no of comment lines are %d ",c);
}
```

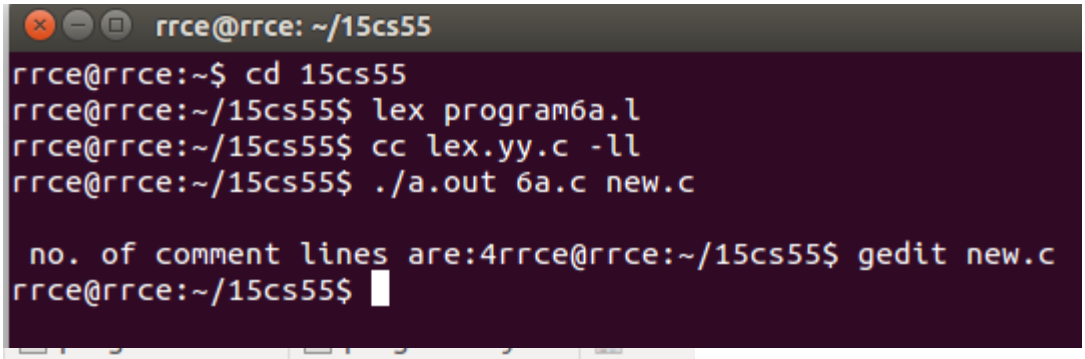
Save the program in **.l** extension.

CREATE AN INPUT FILE (6a.c).

```
#include<stdio.h>  /*header file*/
main() /*main function*/
{
int a=5,b=6;
printf("value of a and b %d%d",a,b); /*print the statement*/
}
/*end of the program*/
```

CREATE AN OUTPUT FILE(new.c).

OUTPUT:




```
#include<stdio.h>
main()
{
int a=5,b=6;
printf("value of a and b %d%d",a,b);
}
```

The output is generated without the comment lines.

PROGRAM 6(B):

Write YACC program to recognize valid identifier, operators and keywords in the given text(C program) file.

LEX PART:

```
%{
 #include<stdio.h>
 #include "y.tab.h"
externyylval;
%}
%%
```

```
[\t];
[+|-|*|/|=|<|>]  {printf("Operator is %s\n",yytext);return OP;}
[0-9]+  {yylval=atoi(yytext);printf("Numbers is %d \n",yylval);return DIGIT;}
int|char|bool|float|void|for|do|while|if|else|return|void  {printf("Keyword is %s
\n",yytext);return KEY;}
[a-zA-Z0-9]+ {printf("Identifier is %s \n",yytext);return ID;}
.;
%%
```

YACC PART:

```
%{
 #include<stdio.h>
 #include<stdlib.h>
int id=0,dig=0,key=0,op=0;
%}
%token DIGIT ID KEY OP
%%
input:
DIGIT input {dig++;}
|ID input {id++;}
|KEY input {key++;}
|OP input {op++;}
|DIGIT {dig++;}
|ID {id++;}
|KEY {key++;}
|OP {op++;}
;
%%
#include<stdio.h>
externintyylex();
```

```
externintyyparse();
extern FILE *yyin;
main()
{
 FILE *myfile = fopen("a.c","r");
if(!myfile)
 {
printf("I cant open sam_input.c !");
return -1;
 }
yyin=myfile;
do{
yyparse();
}while(!feof(yyin));
printf("Numbers = %d \n Keywords = %d \n Identifiers=%d \n Operators=%d
\n",dig,key,id,op);
}
voidyyerror(){
printf("EEk, parse error!Message: ");
exit(-1);
}
```

OUTPUT:

```
cd (directory name)
lex (file name).l
yacc   -d   (file name).y
cc     lex.yy.cy.tab.c   -ll
./a.out
```

CREATE AN INPUT FILE

(“a.c”)

```
rrce@rrce:~/15cs32$ ./a.out
Keyword is void
 Identifier is main
()
{
Keyword is float
 Identifier is a123
;
Keyword is char

Keyword is char
 Identifier is b123
;
Keyword is char

Keyword is if
(Identifier is sum
Operator is =
Operator is =
Numbers is 10
)
 Identifier is printf
("Identifier is Pass
"
Keyword is else

 Identifier is printf
("Identifier is fail
"
}
Numbers = 1
 Keywords = 7
 Identifiers=8
 Operators=2
rrce@rrce:~/15cs32$ lex 1a.l
```

```
void main()
{
float a123;
char a;
char b123;
char c;
if(sum==10)
printf("parse");
else
printf("fail");
}
```

PROGRAM 7:

Design, develop and implement a C/C++/Java program to simulate the working of Shortest remaining time and Round Robin (RR) scheduling algorithms. Experiment with different quantum sizes for RR algorithm

```
#include<stdio.h>
#include<stdlib.h>
struct proc
{
int id;
int arrival;
int burst;
int rem;
int wait;
int finish;
int turnaround;
float ratio;
}process[10];
struct proc temp;
int no;
intchkprocess(int);
intnextprocess();
void roundrobin(int ,int,int[],int[]);
void srtf(int);
main()
{
intn,tq,choice;
intbt[10],st[10],i,j,k;
for(;;)
```

```
{
printf("enter the choice\n");
printf("1.Round Robin\n 2.SRT\n 3.Exit\n");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("Round Robin scheduling algorithm\n");
printf("Enter the no. of process:\n");
scanf("%d",&n);
printf("Enter the burst time for sequences:");
for(i=0;i<n;i++)
{
scanf("%d",&bt[i]);
st[i]=bt[i];
}
printf("enter time quantum:");
scanf("%d",&tq);
roundrobin(n,tq,st,bt);
break;
case 2:
printf("\n \n--SHORTEST REMAINING TIME NEXT--\n\n");
printf("\n \n enter the no. of process:");
scanf("%d",&n);
srtf(n);
break;
case 3:exit(0);
}
}
```

```
}
void roundrobin(intn,inttq,intst[],intbt[])
{
int time=0;
int tat[10],wt[10],i,count=0,swt=0,stat=0,temp1,sq=0,j,k;
float awt=0.0,atat=0.0;
while(1)
{
for(i=0,count=0;i<n;i++)
{
temp1=tq;
if(st[i]==0)
{
count++;
continue;
}
if(st[i]>tq)
st[i]=st[i]-tq;
else
if(st[i]>=0)
{
temp1=st[i];
st[i]=0;
}
sq=sq+temp1;
tat[i]=sq;
}
if(n==count)
break;
```

```
}
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
swt=swt+wt[i];
stat=stat+tat[i];
}
awt=(float)swt/n;
atat=(float)stat/n;
printf("process_no     burst time     wait time    turn around time\n");
for(i=0;i<n;i++)
printf("%d\t\t%d\t\t%d\t\t%d\n",i+1,bt[i],wt[i],tat[i]);
printf("avg wait time is %f\n avgturn around time is %f\n",awt,atat);
}
intchkprocess(int s)
{
int i;
for(i=1;i<=s;i++)
{
if(process[i].rem!=0)
return 1;
}
return 0;
}
intnextprocess()
{
intmin,l,i;
min=32000;
for(i=1;i<=no;i++)
```

```
{
if(process[i].rem!=0&&process[i].rem<min)
{
min=process[i].rem;
l=i;
}
}
return l;
}
void srtf(int n)
{
inti,j,k,time=0;
float tavg,wavg;
for(i=1;i<=n;i++)
{
process[i].id=i;
printf("\n \n enter vthe arrival time for process %d:",i);
scanf("%d",&(process[i].arrival));
printf("enter the burst time for process %d:",i);
scanf("%d",&(process[i].burst));
process[i].rem=process[i].burst;
}
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
{
if(process[i].arrival>process[j].arrival)
{
temp=process[i];
```

```
process[i]=process[j];
process[j]=temp;
}
}
}
no=0;
j=1;
while(chkprocess(n)==1)
{
if(process[no+1].arrival==time)
{
while(process[no+1].arrival==time)
no++;
if(process[j].rem==0)
process[j].finish=time;
j=nextprocess();
}
if(process[j].rem!=0)
{
process[j].rem--;
for(i=1;i<=no;i++)
{
if(i!=j&&process[i].rem!=0)
process[i].wait++;
}
}
else
{
process[j].finish=time;
```

```
j=nextprocess();
time--;
k=j;
}
time++;
}
process[k].finish=time;
printf("\n\n\t\t\t---SHORTEST REMAINING TIME FIRST----");
printf("\n\nProcess Arrival Burst Waiting Finish turnaround Tr/Tb\n");
printf("%5s %9s %7s %10s %8s
%9s\n\n","id","time","time","time","time","time");
for(i=1;i<=n;i++)
{
process[i].turnaround=process[i].wait+process[i].burst;
process[i].ratio=(float)process[i].turnaround/(float)process[i].burst;
printf("%5d %8d %7d %8d %10d %9d
%10.1f",process[i].id,process[i].arrival,process[i].burst,process[i].wait,process[i
].finish,process[i].turnaround,process[i].ratio);
tavg=tavg+process[i].turnaround;
wavg=wavg+process[i].wait;
printf("\n\n");
}
tavg=tavg/n;
wavg=wavg/n;
printf("tavg=%f\t wavg=%f\n",tavg,wavg);
}
```

OUTPUT:

```
cd (directory name)
cc (filename).c
```

./a.out

```
rrce@rrce:~$ cd 15cs55
rrce@rrce:~/15cs55$ cc prog7.c
prog7.c: In function ‘main’:
prog7.c:50:8: warning: incompatible implicit declaration of built-in function
xit’ [enabled by default]
 case 3:exit(0);
        ^
rrce@rrce:~/15cs55$ ./a.out
enter the choice
1.Round Robin
 2.SRT
 3.Exit
1
Round Robin scheduling algorithm
Enter the no. of process:
3
Enter the burst time for sequences:24
3
3
enter time quantum:4
process_no     burst time     wait time     turn around time
1               24              6               30
2               3               4               7
3               3               7               10
avg wait time is 5.666667
 avg turn around time is 15.666667
enter the choice
1.Round Robin
 2.SRT
 3.Exit
2


--SHORTEST REMAINING TIME NEXT--



 enter the no. of process:4


 enter vthe arrival time for process 1:0
```

```
 enter the no. of process:4

 enter vthe arrival time for process 1:0
enter the burst time for process 1:8

 enter vthe arrival time for process 2:1
enter the burst time for process 2:4

 enter vthe arrival time for process 3:2
enter the burst time for process 3:9

 enter vthe arrival time for process 4:3
enter the burst time for process 4:5

                         ---SHORTEST REMAINING TIME FIRST---

Process Arrival Burst Waiting Finish turnaround Tr/Tb
   id       time    time        time     time       time

    1         0       8         9         17         17

    2         1       4         0          5          4

    3         2       9        15         26         24

    4         3       5         2         10          7

tavg=12.699123    wavg=6.500000
enter the choice
1.Round Robin
 2.SRT
 3.Exit
3
rrce@rrce:~/15cs55$
```

PROGRAM 8:

Design, develop and implement a C/C++/Java program to implement Banker's algorithm. Assume suitable input required to demonstrate the results.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int
max[10][10],need[10][10],alloc[10][10],avail[10],completed[10],safeSequence[
10];
intp,r,i,j,process,count;
count=0;
printf("enter the no. of process:");
scanf("%d",&p);
for(i=0;i<p;i++)
completed[i]=0;
printf("\n\n enter the no. of resources:");
scanf("%d",&r);
printf("\n\n enter the max matrix for each process:");
for(i=0;i<p;i++)
{
printf("\nfor process %d:",i+1);
for(j=0;j<r;j++)
scanf("%d",&max[i][j]);
}
printf("\n\n enter the allocation for each process:");
for(i=0;i<p;i++)
{
```

```
printf("\n for process %d:",i+1);
for(j=0;j<r;j++)
scanf("%d",&alloc[i][j]);
}
printf("\n\n enter the available resource:");
for(i=0;i<r;i++)
scanf("%d",&avail[i]);
for(i=0;i<p;i++)
for(j=0;j<r;j++)
need[i][j]=max[i][j]-alloc[i][j];
do
{
printf("\n max matrix :\t allocation matrix:\n");
for(i=0;i<p;i++)
{
for(j=0;j<r;j++)
printf("%d",max[i][j]);
printf("\t\t");
for(j=0;j<r;j++)
printf("%d",alloc[i][j]);
printf("\n");
}

process=-1;
for(i=0;i<p;i++)
{
if(completed[i]==0)
{
process=i;
```

```
for(j=0;j<r;j++)
{
if(avail[j]<need[i][j])
{
process=-1;
break;
}
}
}
if(process!=-1)
break;
}
if(process!=-1)
{
printf("\n process %d runs to completion!",process+1);
safeSequence[count]=process+1;
count++;
for(j=0;j<r;j++)
{
avail[j]+=alloc[process][j];
alloc[process][j]=0;
max[process][j]=0;
completed[process]=1;
}
}
}
while(count!=p && process!=-1);
if(count==p)
{
```

```
printf("\n the system is in a safe state!!\n");
printf("safe sequence:<");
for(i=0;i<p;i++)
printf("%d",safeSequence[i]);
printf(">\n");
}
else

printf("the system is in an unsafe state!!");
}
```

OUTPUT:p

```
rrce@rrce:~$ cd 15cs55
rrce@rrce:~/15cs55$ cc prog8.c
rrce@rrce:~/15cs55$ ./a.out
enter the no. of process:5

 enter the no. of resources:3

 enter the max matrix for each process:
for process 1:7
5
3

for process 2:3
2
2

for process 3:7
0
2

for process 4:2
2
2

for process 5:4
3
3

 enter the allocation for each process:
 for process 1:0
1
0

 for process 2:2
0
0
```

```
 process 2 runs to completion!
 max matrix :    allocation matrix:
753             010
000             000
702             302
222             211
433             002

 process 3 runs to completion!
 max matrix :    allocation matrix:
753             010
000             000
000             000
222             211
433             002

 process 4 runs to completion!
 max matrix :    allocation matrix:
753             010
000             000
000             000
000             000
433             002

 process 1 runs to completion!
 max matrix :    allocation matrix:
000             000
000             000
000             000
000             000
433             002

 process 5 runs to completion!
 the system is in a safe state!!
safe sequence:<23415>
rrce@rrce:~/15cs55$
```

```
 enter the allocation for each process:
 for process 1:0
1
0

 for process 2:2
0
0

 for process 3:3
0
2

 for process 4:2
1
1

 for process 5:0
0
2


 enter the available resource:3
3
2

 max matrix :     allocation matrix:
753              010
322              200
702              302
222              211
433              002

 process 2 runs to completion!
```

PROGRAM 9:

Design, develop and implement a C/C++/Java program to implement page replacement algorithms LRU and FIFO. Assume suitable input required to demonstrate the results.

```
#include<stdio.h>
#include<stdlib.h>
void FIFO(char[],char[],int,int);
void lru(char[],char[],int,int);
void opt(char[],char[],int,int);
int main()
{
intch,YN=1,i,l,f;
char F[10],s[25];

printf("\n\n\t enter the no. of empty frames:");
scanf("%d",&f);
printf("\n\n\tenter the length of the string:");
scanf("%d",&l);
printf("\n\n\tenter the string:");
scanf("%s",s);
for(i=0;i<f;i++)
F[i]=-1;
do
{
printf("\n\n\t********MENU*********");
printf("\n\n\t1:FIFO\n\n2:LRU\n\n3:EXIT");
printf("\n\n\tenter your choice:");
scanf("%d",&ch);
```

```
switch(ch)
{
case 1:
for(i=0;i<f;i++)
{
F[i]=-1;
}
FIFO(s,F,l,f);
break;

case 2:
for(i=0;i<f;i++)
{
F[i]=-1;
}
lru(s,F,l,f);
break;

case 3:
exit(0);
}

printf("\n\n\tDo you want to continue IF YES press 1\n\n\t IF NO press 0:");
scanf("%d",&YN);
}while(YN==1);
return(0);
}
```

```
void FIFO(char s[],char F[],intl,int f)
{
inti,j=0,k,flag=0,cnt=0;
printf("\n\tPAGE\t  FRAMES\t FAULTS");
for(i=0;i<l;i++)
{
for(k=0;k<f;k++)
{
if(F[k]==s[i])
flag=1;
}
if(flag==0)
{
printf("\n\t%c\t",s[i]);
F[j]=s[i];
j++;
for(k=0;k<f;k++)
{
printf("%c",F[k]);
}
printf("\tpage-fault%d",cnt);
cnt++;
}
else
{
flag=0;
printf("\n\t%c\t",s[i]);
for(k=0;k<f;k++)
{
```

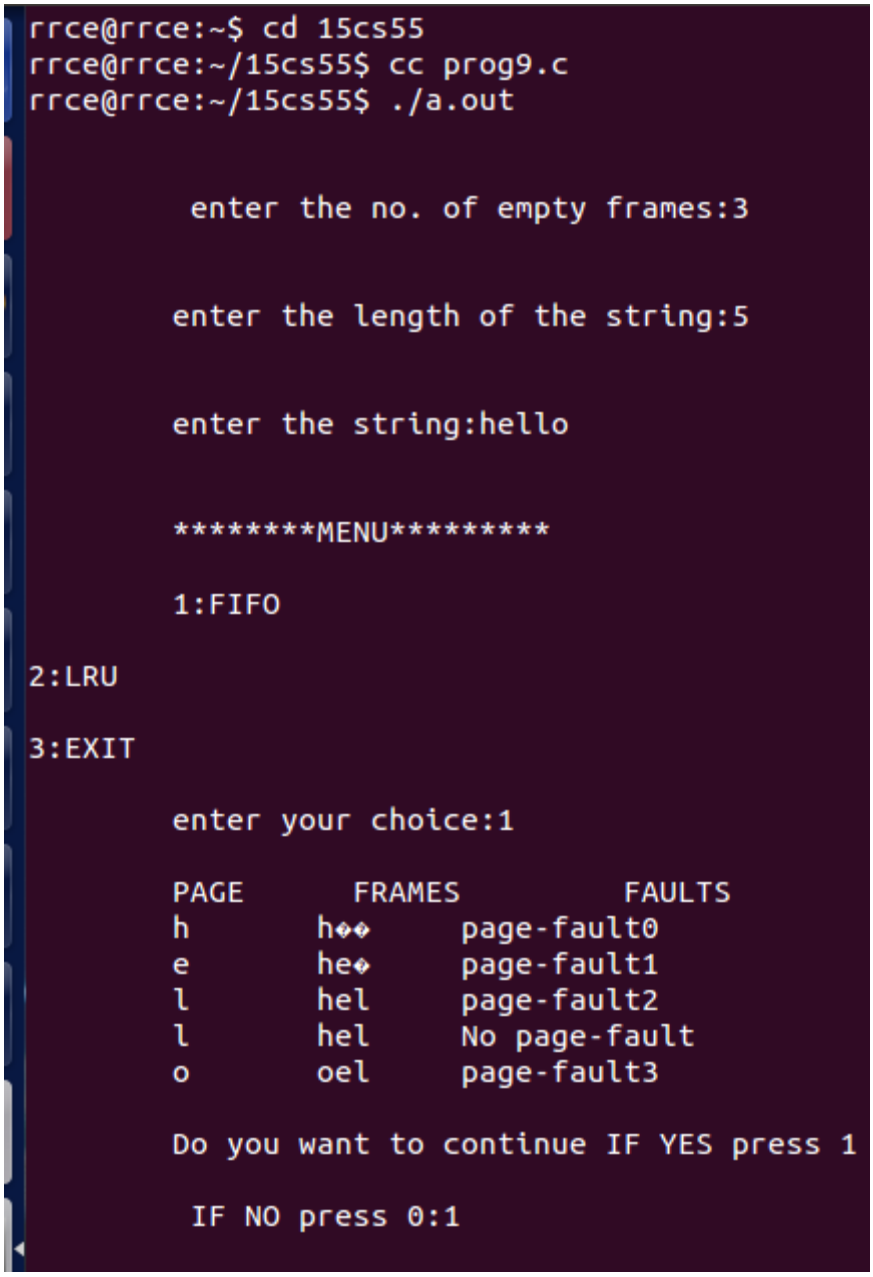
```
printf("%c",F[k]);
}
printf("\tNo page-fault");
}
if(j==f)
j=0;
}
}

void lru(char s[],char F[],intl,int f)
{
inti,j=0,k,m,flag=0,cnt=0,top=0;
printf("\n\t PAGE\t FRAMES\t FAULTS");
for(i=0;i<l;i++)
{
for(k=0;k<f;k++)
{
if(F[k]==s[i])
{
flag=1;
break;
}
}
printf("\n\t%c\t",s[i]);
if(j!=f&&flag!=1)
{
F[top]=s[i];
j++;
if(j!=f)
```

```
top++;
}
else
{
if(flag!=1)
{
for(k=0;k<top;k++)
{
F[k]=F[k+1];
}
F[top]=s[i];
}
if(flag==1)
{
for(m=k;m<top;m++)
{
F[m]=F[m+1];
}
F[top]=s[i];
}
}
for(k=0;k<f;k++)
{
printf("%c",F[k]);
}
if(flag==0)
{
printf("\tPage-fault%d",cnt);
cnt++;
```

```
}
else
printf("\tNo page-fault");
flag=0;
}
}
```

OUTPUT:



```
         IF NO press 0:1


        ********MENU*********

        1:FIFO

2:LRU

3:EXIT

        enter your choice:2

         PAGE    FRAMES  FAULTS
        h       h��     Page-fault0
        e       he�     Page-fault1
        l       hel     Page-fault2
        l       hel     No page-fault
        o       elo     Page-fault3

        Do you want to continue IF YES press 1

         IF NO press 0:1


        ********MENU*********

        1:FIFO

2:LRU

3:EXIT

        enter your choice:3
```