**PROGRAM: 1**

1. **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 & operators present and print them separately.**

%{

#include<stdio.h>

int id=0,op=0,br=0;

%}

%%

[a-zA-Z0-9]+ {id++; printf("\nthe identifier is:\n"); ECHO;}

[+|\*|/]+ {op++; printf("\nthe operator is :\n"); ECHO;}

[-] {op++; printf("\nthe operator is :\n"); ECHO;}

"(" {br++;}

")" {if(br==0) {br=1; return 0;} else br--;}

%%

int main()

{

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

yylex();

if((op+1)==id && br==0)

{

printf("the entered expression is valid\n");

printf("the total number of identifiers %d:\n",id);

printf("the total number of operators %d:\n",op);

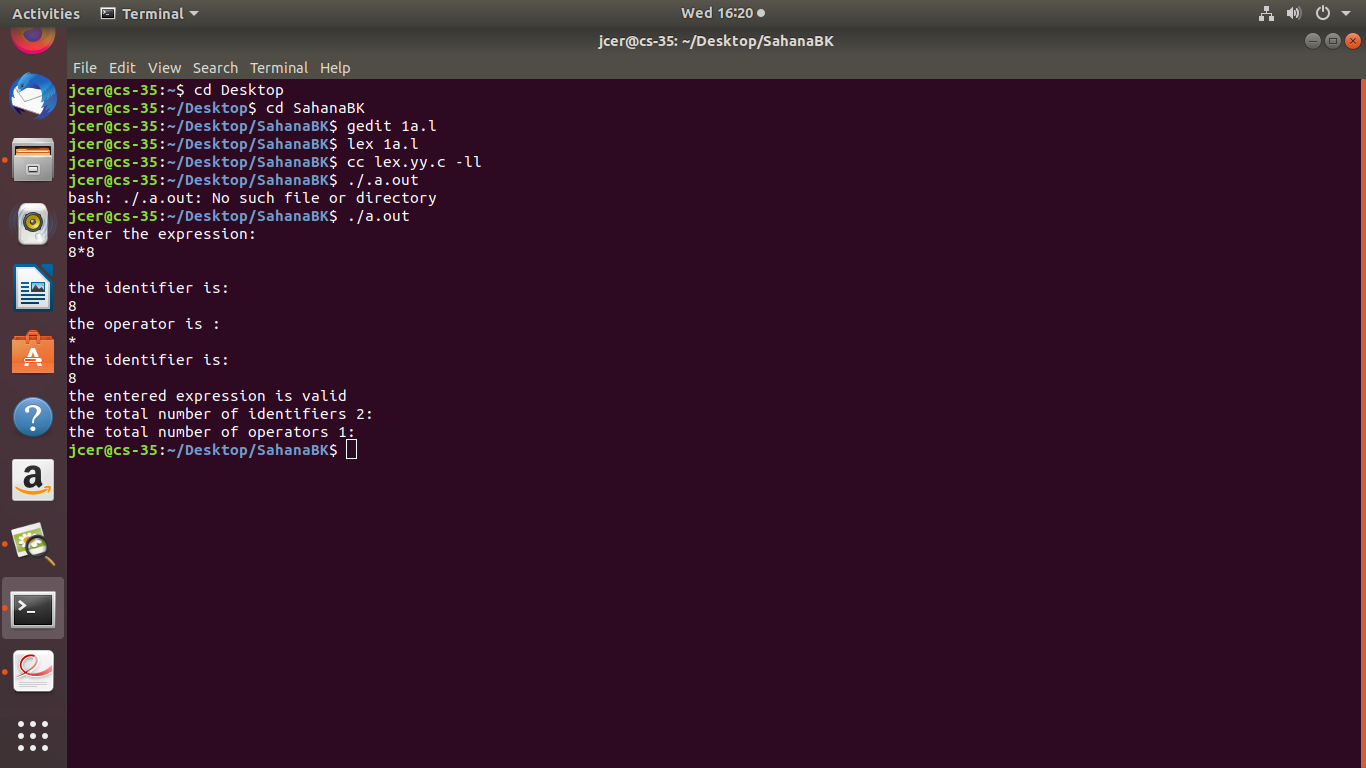
}

else

printf("the entered expression is invalid\n");

}

**OUTPUT:**



1. **Write YACC program to evaluate arithmetic expression involving operators:**

**-, \*, and /**

//lex program

%{

#include "y.tab.h"

extern int yylval;

%}

%%

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

. {return yytext[0];}

%%

//yacc program

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token NUM

%left '+''-'

%left '\*''/'

%nonassoc UMINUS

%%

exp:NUM{$$=$1; yylval=$$;}

|exp'+'exp {$$=$1+$3;yylval=$$;}

|exp'-'exp {$$=$1-$3;yylval=$$;}

|exp'\*'exp {$$=$1\*$3;yylval=$$;}

|exp'/'exp {if($3==0) {printf("divided by zero error\n"); exit(0);}

else $$=$1/$3; yylval=$$;}

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

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

;

%%

int main()

{

printf("enter the airthmatic expressionto evaluate:\n");

yyparse();

printf("valid expression\n");

printf("result=%d\n",yylval);

exit(0);

}

yyerror()

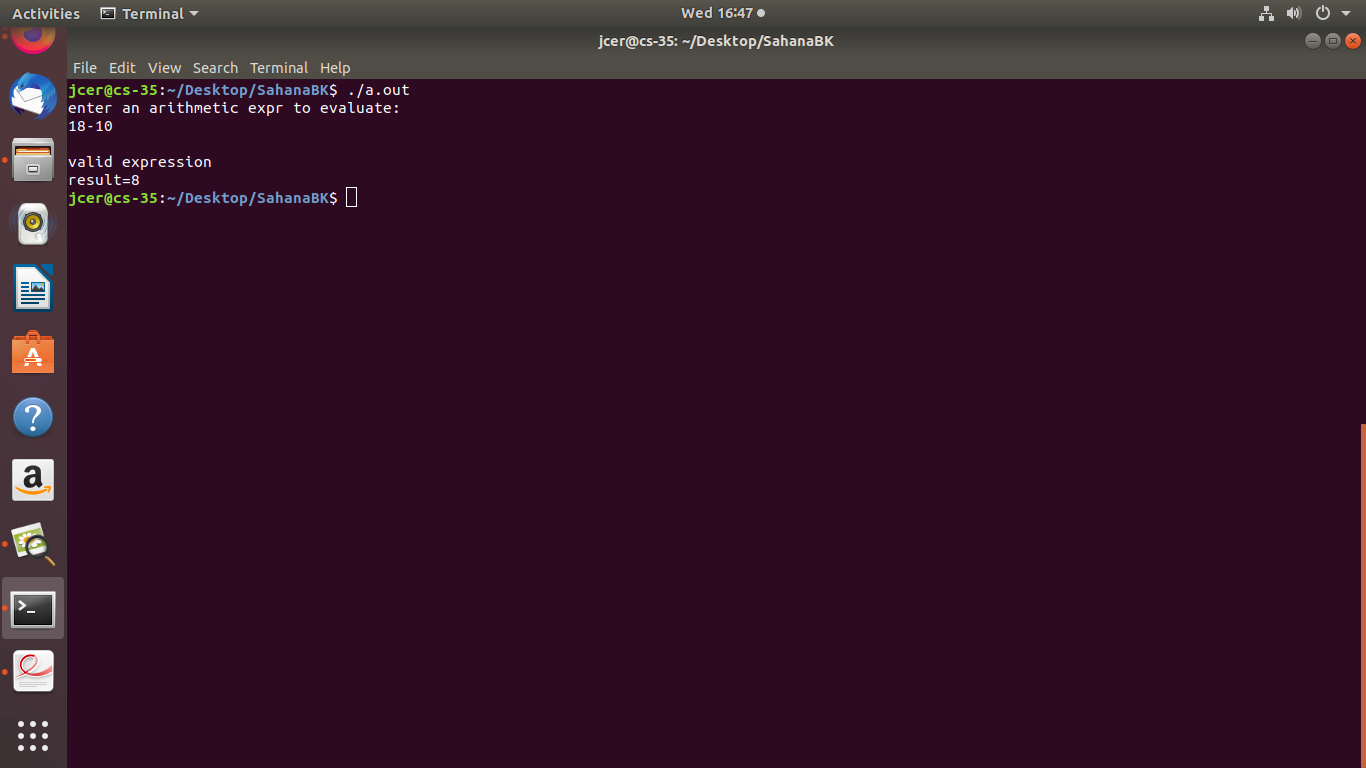
{

printf("invalid expression\n");

exit(0);

}

**OUTPUT:**



**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 an b (note: input n value)**

//lex program//

%{

#include "y.tab.h"

%}

%%

[a] {return A;}

[b] {return B;}

. {return yytext[0];}

%%

//YACC program//

%{

#include<stdio.h>

#include<stdlib.h>

int valid;

%}

%token A B

%%

str: A expr B | B

expr: A expr|;

%%

int main()

{

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

yyparse();

printf("valid grammer\n");

exit(0);

}

yyerror()

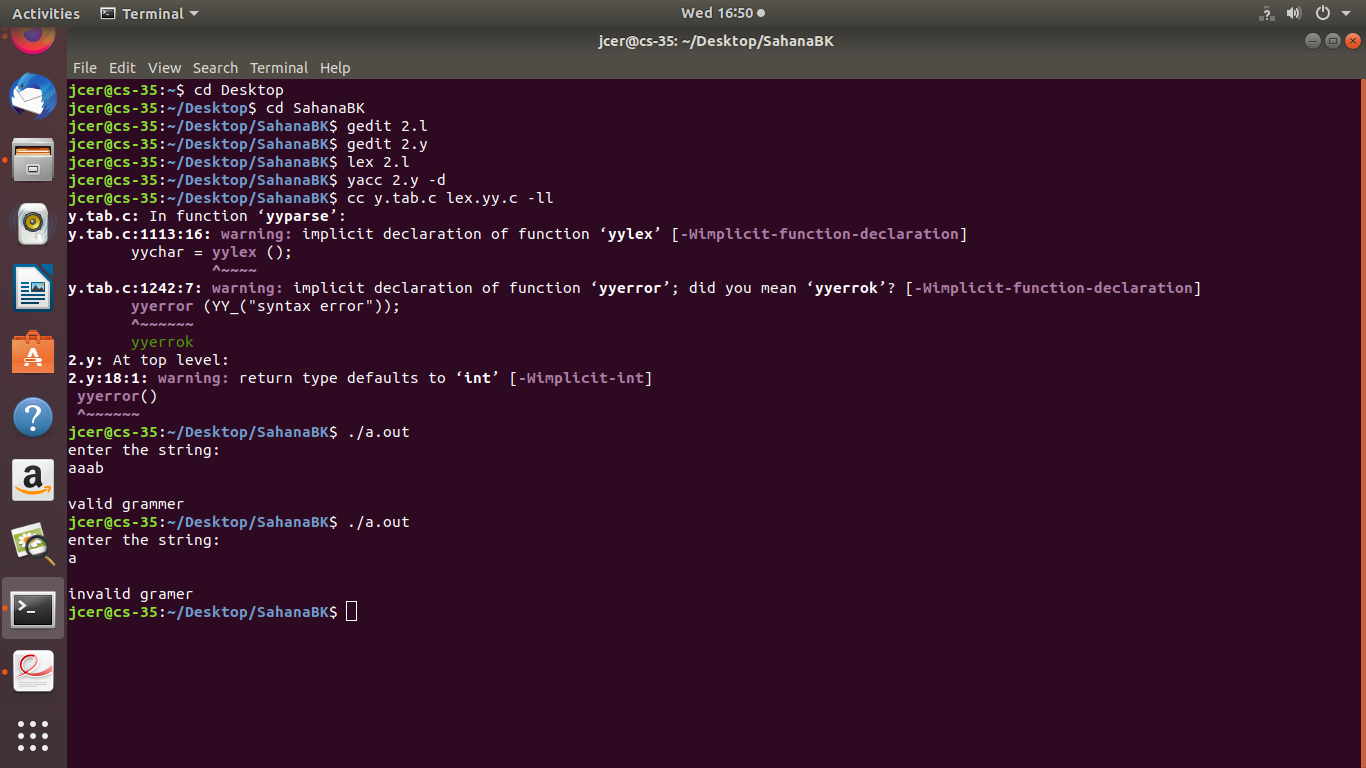
{

printf("invalid gramer\n");

exit(0);

}

**OUTPUT:**

****

**PROGRAM: 3**

**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<stdlib.h>

#include<string.h>

#include<stdio.h>

char prod[3][10]={"A->aBa", "B->bB", "B->@"}; //Note - @ is considered to be the empty string.

char input[10], stack[25];

int top=-1;

int j=0,k,l;

//Push function for the stack

void push(char item)

{

stack[++top]=item;

}

//Pop function for the stack

void pop()

{

top=top-1;

}

//Function to display the stack

void display()

{

int j;

for(j=top;j>=0;j--)

printf("%c",stack[j]);

}

void stackpush(char p)

{

if(p=='A')

{

pop();

for(j=strlen(prod[0])-1; j>=3; j--)

push(prod[0][j]);

}

else

{

pop();

for(j=strlen(prod[1])-1; j>=3; j--)

push(prod[1][j]);

}

}

void main(){

char c;

int i;

printf("Enter the input string terminated with $ to parse: ");

scanf("%s",&input);

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

{

if((input[i]!='a')&&(input[i]!='b')&&(input[i]!='$'))

{

printf("Invalid String.\n");

exit(0);

}

}

if(input[i-1]!='$')

{

printf("\n Input String is enetred without the end marker $.\n");

exit(0);

}

push('$');

push('A');

i=0;

printf("\n\n");

printf("Stack\tInput\tAction\n");

while(i!=strlen(input) && stack[top]!='$')

{

printf("\n");

for(l=top; l>=0; l--)

printf("%c",stack[l]);

printf("\t");

for(l=i; l<strlen(input); l++)

printf("%c",input[l]);

printf("\t");

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

{

printf("A->aBa");

stackpush('A');

}

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

{if(input[i]!='b')

{

printf("B->@");

printf("\t Matched @");

pop();

}

else

{

printf("B->bB");

stackpush('B');

}

}

else

{

if(stack[top]==input[i])

{

printf("Pop %c",input[i]);

printf("\t Matched %c",input[i]);

pop();

i++;

}

else

break;

}

}

if(stack[top]=='$' && input[i]=='$')

{

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

printf("\nThe string is VALID and it is ACCEPTED.\n");

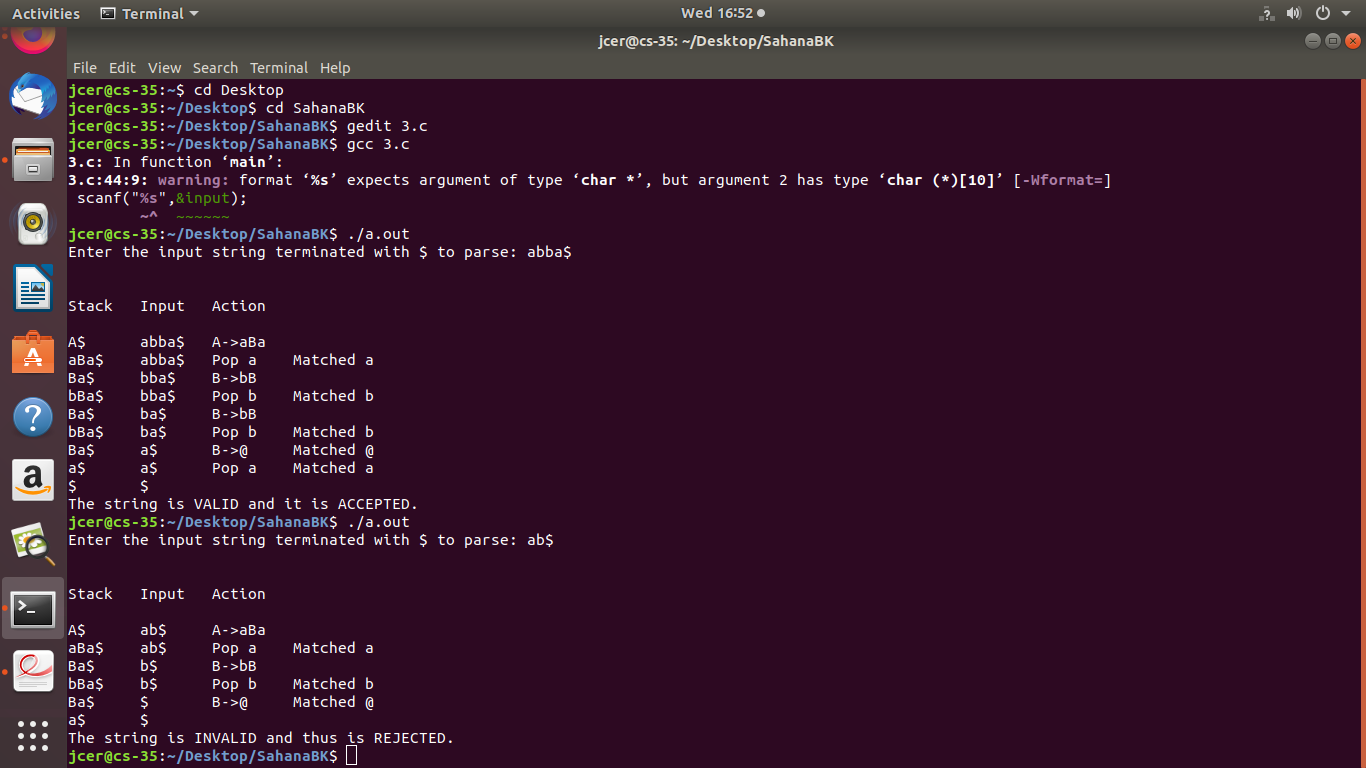
}

else

printf("\nThe string is INVALID and thus is REJECTED.\n");

}

**OUTPUT:**



**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();

int main()

{

puts("GRAMMAR is \n E->E+E \n E->E\*E \n E->(E) \n E->id");

puts("\nEnter input string :");

gets(a);

c=strlen(a);

strcpy(act,"SHIFT->");

puts("stack \t input \t action");

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%s symbols\n",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+2]='\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';

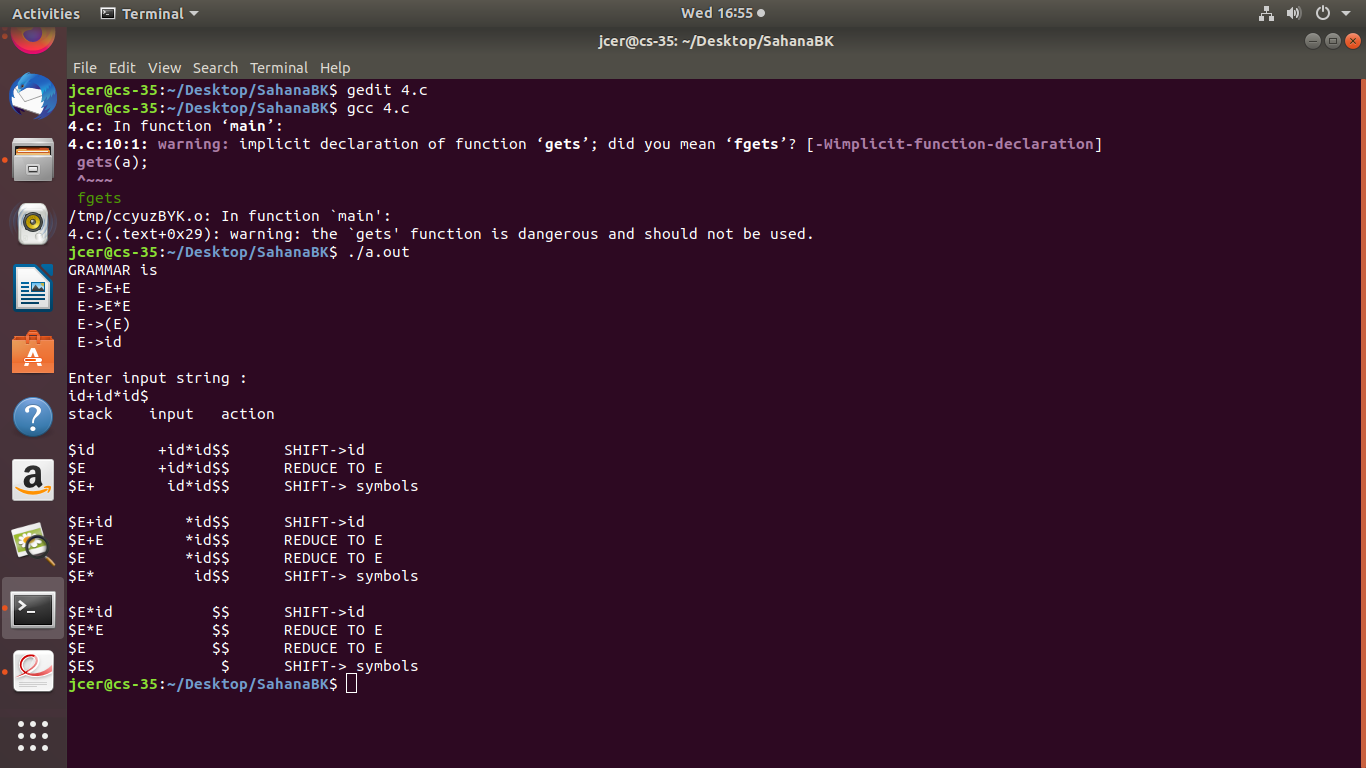
printf("\n$%s\t%s$\t%s",stk,a,ac);

i=i-2;

}

}

**OUTPUT:**



**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>

#include<string.h>

char op[2], arg1[5], arg2[5], result[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", result,arg1,op,arg2);

if(strcmp(op,"+")==0)

{

fprintf(fp2,"\nMOV R0,%s",arg1);

fprintf(fp2,"\nADD R0,%s",arg2);

fprintf(fp2,"\nMOV %s,R0",result);

}

if(strcmp(op,"\*")==0)

{

fprintf(fp2,"\nMOV R0,%s",arg1);

fprintf(fp2,"\nMUL R0,%s",arg2);

fprintf(fp2,"\nMOV %s,R0",result);

}

if(strcmp(op,"-")==0)

{

fprintf(fp2,"\nMOV R0,%s",arg1);

fprintf(fp2,"\nSUB R0,%s",arg2);

fprintf(fp2,"\nMOV %s,R0",result);

}

if(strcmp(op,"=")==0)

{

fprintf(fp2,"\nMOV R0,%s",arg1);

fprintf(fp2,"\nMOV %s,R0",result);

}

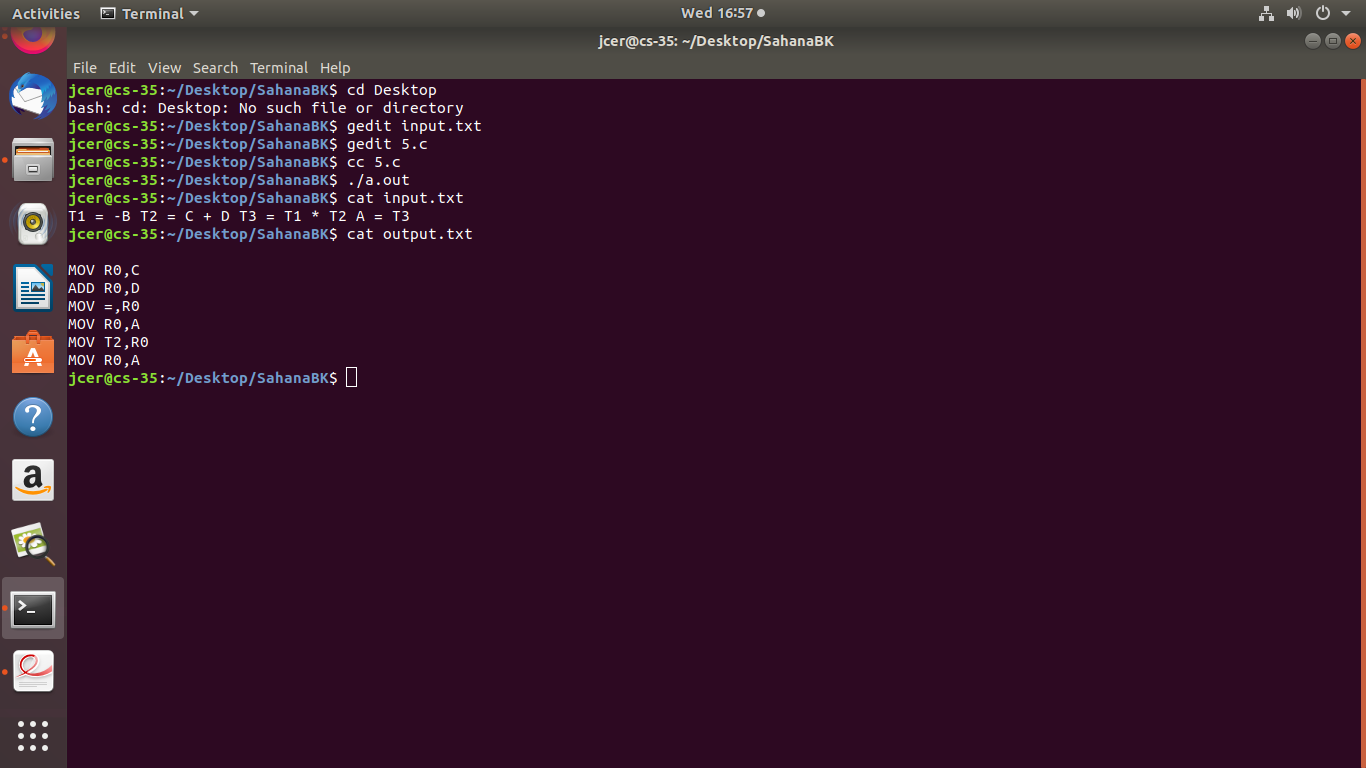
}

fclose(fp1);

fclose(fp2);

}

**OUTPUT:**

****

**PROGRAM: 6**

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

%{

#include<stdio.h>

#include<stdlib.h>

int com=0;

%}

%%

"/\*"[^\n]+"\*/" {com++;}

"/\*/\*"[^\n]+"/\*/\*" {com++;}

"//"[^\n]+"//" {com++;}

%%

int main()

{

yyin=fopen("1.c","r");

yyout=fopen("2.c","w");

yylex();

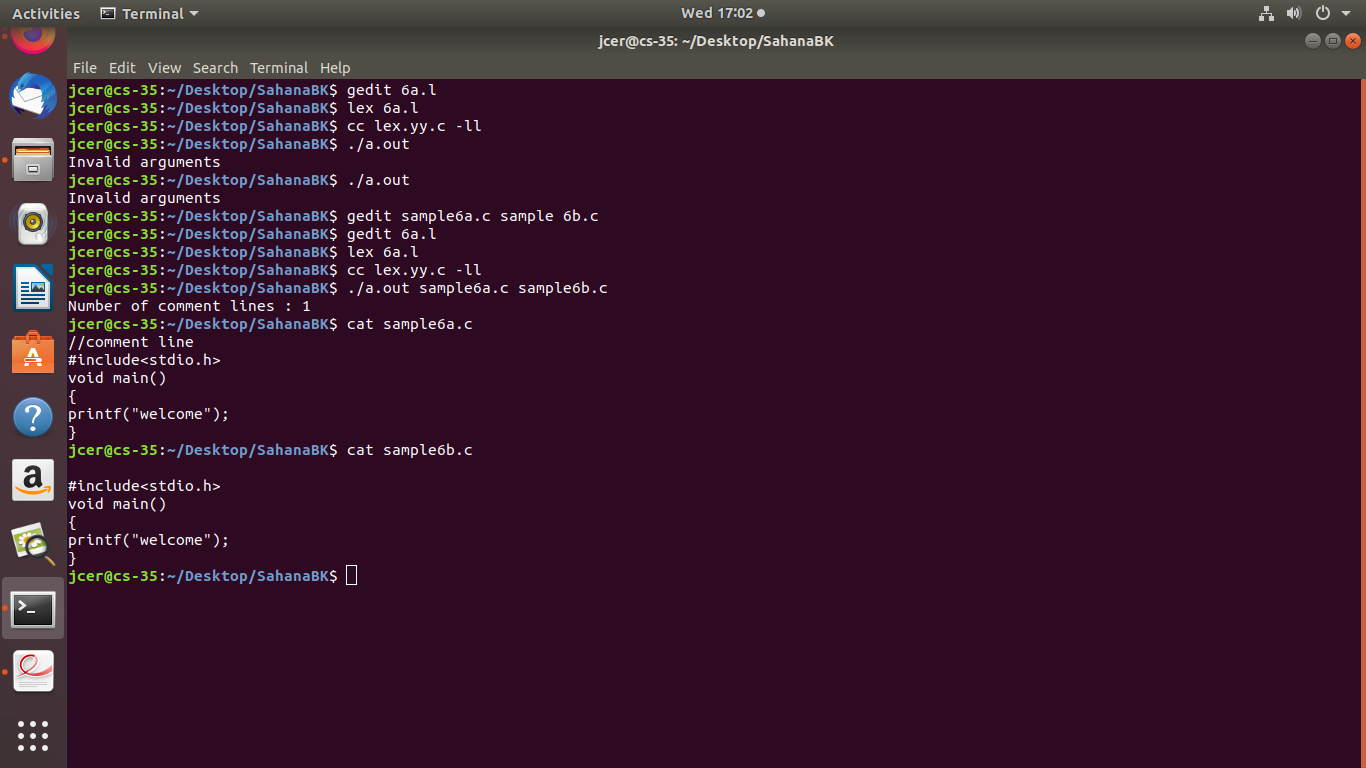
fclose(yyin);

fclose(yyout);

printf("number of comments lines in the given file:%d\n",com);

}

**OUTPUT:**



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

//lex program

%{

#include<stdio.h>

#include"y.tab.h"

%}

%%

int|char|bool|float|void|for|do|while|if|else|return|main {printf("keyword is %s\n",yytext); return KEY;}

[+|-|\*|/|=|<|>] {printf("operator is %s\n",yytext); return OP;}

[a-zA-Z][a-zA-Z0-9]\* {printf("identifire is %s\n",yytext); return ID;}

.;

%%

//yacc program

%{

#include<stdio.h>

#include<stdlib.h>

int id=0,key=0,op=0;

%}

%token ID KEY OP

%%

input:ID input{id++;}

| KEY input{key++;}

| OP input{op++;}

| ID {id++;}

| KEY {key++;}

| OP {op++;}

;

%%

extern FILE \*yyin;

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

{

yyin=fopen(argv[1],"r");

yyparse();

printf("keywords=%d\n identifires=%d\n operators=%d\n",key,id,op);

}

void yyerror()

{

printf("error");

exit(0);

}

**OUTPUT:**

**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.**

**/\* The working of Shortest remaining time \*/**

#include<stdio.h>

int main()

{

int at[10],bt[10],temp[10];

int i,smallest,count=0,time,n;

double wt=0,tt=0,end;

float awt,att;

printf("enter the total number of processes:\n");

scanf("%d",&n);

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

{

printf("enter arrival time:\n");

scanf("%d", &at[i]);

printf("enter burst time:\n");

scanf("%d",&bt[i]);

temp[i]=bt[i];

}

bt[9]=9999;

for(time=0;count!=n;time++)

{

smallest=9;

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

{

if(at[i]<=time &&bt[i] < bt[smallest]&&bt[i]>0)

{

smallest=i;

}

}

bt[smallest]--;

if(bt[smallest]==0)

{

count++;

end=time+1;

wt=wt+end-at[smallest]-temp[smallest];

tt=tt+end-at[smallest];

}

}

awt=wt/n;

att=tt/n;

printf("Average waiting time:%f\n",awt);

printf("Average turnarround time:%f\n",att);

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sahana@Sahana:~/Desktop/lab$ cc 7a.c

Sahana@Sahana:~/Desktop/lab$ ./a.out

enter the total number of processes:

4

enter arrival time:

5

enter burst time:

6

enter arrival time:

3

enter burst time:

2

enter arrival time:

4

enter burst time:

8

enter arrival time:

9

enter burst time:

6

Average waiting time:3.750000

Average turnarround time:9.250000

**/\* Round Robin scheduling algorithm \*/**

#include<stdio.h>

int main()

{

int i,count=0,n,temp1,tq,sq=0;

int st[20],tat[20],wt[20],bt[20];

float swt=0,stat=0;

double awt=0,atat=0;

printf("enter the total number of processes:\n");

scanf("%d",&n);

printf("enter the brust time:\n");

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

{

scanf("%d",&bt[i]);

st[i]=bt[i];

}

printf("enter the time quantum:\n");

scanf("%d",&tq);

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;

}

printf("Process\_no Brust time Wait time Turn arround time\n");

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

{

wt[i]=tat[i]-bt[i];

swt=swt+wt[i];

stat=stat+tat[i];

printf("%d\t\t%d\t%d\t%d\n",i+1,bt[i],wt[i],tat[i]);

}

awt=(float)swt/n;

atat=(float)stat/n;

printf("Avg Wait time is %f\nAvg turn arround time is %f\n",awt,atat);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sahana@Sahana:~/Desktop/lab$ cc 7b.c

Sahana@Sahana:~/Desktop/lab$ ./a.out

enter the total number of processes:

4

enter the brust time:

10

3

6

5

enter the time quantum:

3

Process\_no Brust time Wait time Turn arround time

1 10 14 24

2 3 3 6

3 6 12 18

4 5 15 20

Avg Wait time is 11.000000

Avg turn arround time is 17.000000

**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 available[10],allocation[10][10],maxi[10][10],need[10][10],c[10],i,j,k,l=0,n,r,seq[10];

int grant()

{

int x;

for(x=0;x<r;x++)

{

if(!(need[i][x]<=available[x]))

return 0;

}

return 1;

}

void insert(int ele)

{

seq[l++]=ele;

}

void display()

{

printf("Safe sequence:<");

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

printf("%d,",seq[i]);

printf("%d>\n",seq[i]);

}

void main()

{

int sum,p[10],z=0,allow;

printf("Enter no. of processes:");

scanf("%d",&n);

printf("Enter no. of resources:");

scanf("%d",&r);

printf("Enter available resource matrix:\n");

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

scanf("%d",&available[i]);

printf("Enter allocation matrix:\n");

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

{

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

{

scanf("%d",&allocation[i][j]);

}

}

printf("Enter max. matrix:\n");

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

{

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

{

scanf("%d",&maxi[i][j]);

need[i][j]=maxi[i][j]-allocation[i][j];

}

}

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

{

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

{

allow=grant();

if(allow!=0 && p[i]!=1)

{

insert(i);

p[i]=1;

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

{

available[j]=available[j]+allocation[i][j];

}

}

}

}

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

{

if(p[i]==0)

{

printf("Unsafe state\n");

exit(0);

}

}

printf("Safe state\n");

display();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sahana@Sahana:~/Desktop/lab$ gedit 8.c

Sahana@Sahana:~/Desktop/lab$ cc 8.c

Sahana@Sahana:~/Desktop/lab$ ./a.out

Enter no. of processes:5

Enter no. of resources:3

Enter available resource matrix:

6

3

2

Enter allocation matrix:

2

0

1

3

6

5

4

9

2

0

3

2

5

0

1

Enter max. matrix:

4

3

6

2

5

0

2

2

3

2

3

3

3

6

6

Safe state

Safe sequence:<1,2,3,4,0,0>

**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.**

**/\* LRU \*/**

#include<stdio.h>

int main()

{

int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];

printf("Enter no of pages:");

scanf("%d",&n);

printf("Enter the refrences string:");

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

scanf("%d",&p[i]);

printf("Enter no of frames:");

scanf("%d",&f);

p[i]-f;

q[k]=p[k];

printf("\n\t%d\n",q[k]);

c++;

k++;

for(i=1;i<n;i++)

{

c1=0;

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

{

if(p[i]!=q[j])

c1++;

}

if(c1==f)

{

c++;

if(k<f)

{

q[k]=p[i];

k++;

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

printf("\t%d",q[j]);

printf("\n");

}else

{

for(r=0;r<f;r++)

{

c2[r]=0;

for(j=i-1;j<n;j--)

{

if(q[r]!=p[j])

c2[r]++;

else

break;

}

}

for(r=0;r<f;r++)

b[r]=c2[r];

for(r=0;r<f;r++)

{

for(j=r;j<f;j++)

{

if(b[r]<b[j])

{

t=b[r];

b[r]=b[j];

b[j]=t;

}

}

}

for(r=0;r<f;r++)

{

if(c2[r]==b[0])

q[r]=p[i];

printf("\t%d",q[r]);

}

printf("\n");

}

}

}

printf("\n The no of page faults is %d",c);

}

\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sahana@Sahana:~/Desktop/lab$ cc 9b.c

Sahana@Sahana:~/Desktop/lab$ ./a.out

Enter no of pages:12

Enter the refrences string:

1

2

3

4

1

2

5

1

2

3

2

5

Enter no of frames:3

1

1 2

1 2 3

4 2 3

4 1 3

4 1 2

5 1 2

3 1 2

3 5 2

The no of page faults is 9

**/\* FIFO \*/**

#include<stdio.h>

int main()

{

int i,j,n,a[50],frame[10],no,k,avail,count=0,h;

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

scanf("%d",&n);

printf("Enter the page number\n");

for(i=1;i<=n;i++)

scanf("%d",&a[i]);

printf("Enter the number of frames:");

scanf("%d",&no);

for(i=0;i<no;i++)

frame[i]=-1;

j=0;

printf("ref string\t page frames\n");

for(i=1;i<=n;i++)

{

printf("%d\t\t",a[i]);

avail=0;

for(k=0;k<no;k++)

if(frame[k]==a[i])

avail=1;

if(avail==0)

{

frame[j]=a[i];

j=(j+1)%no;

count++;

for(k=0;k<no;k++)

printf("%d\t",frame[k]);

}

printf("\n");

}

printf("page fault is %d",count);

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sahana@Sahana:~/Desktop/lab$ cc 9a.c

Sahana@Sahana:~/Desktop/lab$ ./a.out

Enter the number of pages:

15

Enter the page number

7

0

1

2

0

3

0

4

2

3

0

3

1

2

0

Enter the number of frames:3

ref string page frames

7 7 -1 -1

0 7 0 -1

1 7 0 1

2 2 0 1

0

3 2 3 1

0 2 3 0

4 4 3 0

2 4 2 0

3 4 2 3

0 0 2 3

3

1 0 1 3

2 0 1 2

0

page fault is 12