

ANNAMALAI UNIVERSITY
FACULTY OF ENGINEERING & TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING

Bachelor of Engineering (B.E.)

COEP-707 COMPILER DESIGN & NETWORKS
LAB MANUAL

LIST OF EXPERIMENTS

EX. No.	DESCRIPTION	PAGE NO.
CYCLE – I (COMPILER DESIGN)		
1	IMPLEMENTATION OF LEXICAL ANALYZER FOR IF STATEMENT	3
2	IMPLEMENTATION OF LEXICAL ANALYZER FOR ARITHMETIC EXPRESSION	7
3	IMPLEMENTATION OF LEXICAL ANALYZER USING LEXTOOL	11
4	CONSTRUCTION OF NFA FROM REGULAR EXPRESSION	16
5	CONSTRUCTION OF DFA FROM NFA	22
6	IMPLEMENTATION OF SHIFT REDUCE PARSING ALGORITHM	28
7	IMPLEMENTATION OF OPERATOR PRECEDENCE PARSER	32
8	IMPLEMENTATION OF RECURSIVE DESCENT PARSER	36
9	IMPLEMENTATION OF CODE OPTIMIZATION TECHNIQUES	41
10	IMPLEMENTATION OF CODE GENERATOR	45
CYCLE – II (NETWORKS)		
11	(A) TO FIND THE IP ADDRESS OF LOCAL HOST	49
	(B) TO FIND THE IP ADDRESS OF REMOTE HOST	51
12	IMPLEMENTATION OF ECHO SERVER AND CLIENT USING TCP SOCKETS	53
13	IMPLEMENTATION OF ECHO SERVER AND CLIENT USING UDP SOCKETS	57
14	SEND AND RECEIVE MESSAGE BETWEEN CLIENT AND SERVER USING TCP	60
15	SEND AND RECEIVE MESSAGE BETWEEN CLIENT AND SERVER USING UDP	63
16	SLIDING WINDOW PROTOCOLS	66

Ex. No: 1

Date:

Implementation of Lexical Analyzer for 'if' Statement

Aim:

To write a C program to implement lexical analyzer for 'if' statement.

Algorithm:

Input: Programming language 'if' statement

Output: A sequence of tokens.

Tokens have to be identified and its respective attributes have to be printed.

Lexeme

Token

If	<1,1>
variable-name	<2,#address>
numeric-constant	<3,#address>
;	<4,4>
(<5,0>
)	<5,1>
{	<6,0>
}	<6,1>
>	<62,62>
>=	<620,620>
<	<60,60>
<=	<600,600>
!	<33,33>
!=	<330,330>
=	<61,61>
==	<610,610>

Program:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<string.h>
char vars[100][100];
int vcnt;
char input[1000],c;
char token[50],tlen;
int state=0,pos=0,i=0,id;
```

```

char*getAddress(char str[])
{
for(i=0;i<vcnt;i++)
if(strcmp(str,vars[i])==0)
return vars[i];
strcpy(vars[vcnt],str);
return vars[vcnt++];
}
intisrelop(char c)
{
if(c=='>'||c=='<'||c=='||c=='=')
return 1;
else
return 0;
}
int main(void)
{
clrscr();
printf("Enter the Input String:");
gets(input);
do
{
c=input[pos];
putchar(c);
switch(state)
{
case 0:
if(c=='i')
state=1;
break;
case 1:
if(c=='f')
{
printf("\t<1,1>\n");
state =2;
}
break;
case 2:
if(isspace(c))
printf("\b");
if(isalpha(c))
{
token[0]=c;
tlen=1;
state=3;
}
if(isdigit(c))
state=4;

```

```

if(isrelop(c))
state=5;
if(c==';')printf("\t<4,4>\n");
if(c=='()')printf("\t<5,0>\n");
if(c=='')printf("\t<5,1>\n");
if(c=='{' ) printf("\t<6,1>\n");
if(c=='}' ) printf("\t<6,2>\n");
break;
case 3:
if(!isalnum(c))
{
token[tlen]='\0';
printf("\b\t<2,%p>\n",getAddress(token));
state=2;
pos--;
}
else
token[tlen++]=c;
break;
case 4:
if(!isdigit(c))
{
printf("\b\t<3,%p>\n",&input[pos]);
state=2;
pos--;
}
break;
case 5:
id=input[pos-1];
if(c=='=')
printf("\t<%d,%d>\n",id*10,id*10);
else
{
printf("\b\t<%d,%d>\n",id,id);
pos--;
}
state=2;
break;
}
pos++;
}
while(c!=0);
getch();
return 0;
}

```

Sample Input & Output:

Enter the input string: if(a>=b) max=a;

if	<1,1>
(<5,0>
a	<2,0960>
>=	<620,620>
b	<2,09c4>
)	<5,1>
max	<2,0A28>
=	<61,61>
a	<2,0A8c>
;	<4,4>

Result:

The above C program was successfully executed and verified.

Ex. No: 2

Date:

Implementation of Lexical Analyzer for Arithmetic Expression

Aim:

To write a C program to implement lexical analyzer for Arithmetic Expression.

Algorithm:

Input: Programming language arithmetic expression

Output: A sequence of tokens.

Tokens have to be identified and its respective attributes have to be printed.

Lexeme *****	Token *****
Variable name	<1,#address>
Numeric constant	<2,#address>
;	<3,3>
=	<4,4>
+	<43,43>
+=	<430,430>
-	<45,45>
-=	<450,450>
*	<42,42>
*=	<420,420>
/	<47,47>
/=	<470,470>
%	<37,37>
%=	<370,370>
^	<94,94>
^=	<940,940>

Program:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<string.h>
char vars[100][100];
int vcnt;
char input[1000],c;
char token[50],tlen;
int state=0,pos=0,i=0,id;
```

```

char *getAddress(char str[])
{
for(i=0;i<vcnt;i++)
if(strcmp(str,vars[i])==0)
return vars[i];
strcpy(vars[vcnt],str);
return vars[vcnt++];
}
intisrelop(char c)
{
if(c=='+'||c=='-'||c=='*'||c=='/'||c=='%'||c=='^')
return 1;
else
return 0;
}
int main(void)
{
clrscr();
printf("Enter the Input String:");
gets(input);
do
{
c=input[pos];
putchar(c);
switch(state)
{
case 0:
if(isspace(c))
printf("\b");
if(isalpha(c))
{
token[0]=c;
tlen=1;
state=1;
}
if(isdigit(c))
state=2;
if(isrelop(c))
state=3;
if(c==';')
printf("\t<3,3>\n");
if(c=='=')
printf("\t<4,4>\n");
break;

```



```

case 1:
if(!isalnum(c))
{
token[tlen]='\0';
printf("\b\t<1,%p>\n",getAddress(token));
state=0;
pos--;
}
else
token[tlen++]=c;
break;
case 2:
if(!isdigit(c))
{
printf("\b\t<2,%p>\n",&input[pos]);
state=0;
pos--;
}
break;
case 3:
id=input[pos-1];
if(c=='=')
printf("\t<%d,%d>\n",id*10,id*10);
else
{
printf("\b\t<%d,%d>\n",id,id);
pos--;
}
state=0;
break;
}
pos++;
}
while(c!=0);
getch();
return 0;
}

```

Sample Input & Output:

Enter the Input String: a=a*2+b/c;

a	<1,08CE>
=	<4,4>
a	<1,08CE>
*	<42,42>
2	<2,04E9>
+	<43,43>
b	<1,0932>
/	<47,47>
c	<1,0996>
;	<3,3>

Result:

The above C program was successfully executed and verified.

Ex. No: 3

Date:

Implementation of Lexical Analyzer using Lex Tool

Aim:

To write a C program to implement Lexical Analyzer using Lex Tool.

Algorithm:

1. Start the program.
2. Lex program consists of three parts.
 - a. Declaration %%
 - b. Translation rules %%
 - c. Auxiliary procedure.
3. The declaration section includes declaration of variables, maintest, constants and regular definitions.
4. Translation rule of lex program are statements of the form
 - a. P1 {action}
 - b. P2 {action}
 - c. ...
 - d. ...
 - e. Pn {action}
5. Write a program in the vi editor and save it with .l extension.
6. Compile the lex program with lex compiler to produce output file as
lex.yy.c. eg \$ lex filename.l
\$ cc lex.yy.c -ll
7. Compile that file with C compiler and verify the output.

Program:

Lexical.C:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
void main()
{
FILE *fi,*fo,*fop,*fk;
int flag=0,i=1;
char c,t,a[15],ch[15],file[20];
clrscr();
printf("\n Enter the File Name:");
scanf("%s",&file);
fi=fopen(file,"r");
```

```

fo=fopen("inter.c","w");
fop=fopen("oper.c","r");
fk=fopen("key.c","r");
c=getc(fi);
while(!feof(fi))
{
if(isalpha(c)||isdigit(c)||(c=='['||c==']'||c=='.'==1))
fputc(c,fo);
else
{
if(c=='\n')
fprintf(fo,"\t$\t");
else
fprintf(fo,"\t%c\t",c);
}
c=getc(fi);
}
fclose(fi);
fclose(fo);
fi=fopen("inter.c","r");
printf("\n Lexical Analysis");
fscanf(fi,"%s",a);
printf("\n Line: %d\n",i++);
while(!feof(fi))
{
if(strcmp(a,"$")==0)
{
printf("\n Line: %d \n",i++);
fscanf(fi,"%s",a);
}
fscanf(fop,"%s",ch);
while(!feof(fop))
{
if(strcmp(ch,a)==0)
{
fscanf(fop,"%s",ch);
printf("\t\t%s\t\t\t%s\n",a,ch);
flag=1;
}
fscanf(fop,"%s",ch);
}
rewind(fop);
fscanf(fk,"%s",ch);
while(!feof(fk))
{
if(strcmp(ch,a)==0)
{

```

```

fscanf(fk,"%k",ch);
printf("\t\t%s\t:\tKeyword\n",a);
flag=1;
}
fscanf(fk,"%s",ch);
}
rewind(fk);
if(flag==0)
{
if(isdigit(a[0]))
printf("\t\t%s\t:\tConstant\n",a);
else
printf("\t\t%s\t:\tIdentifier\n",a);
}
flag=0;
fscanf(fi,"%s",a);
}
getch();
}

```

Key.C:

```

int
void
main
char
if
for
while
else
printf
scanf
FILE
include
stdio.h
conio.h
iostream.h

```

Oper.C:

```

( open para
) closepara
{ openbrace
} closebrace
< lesser
> greater
" doublequote
' singlequote

```

: colon
; semicolon
preprocessor
= equal
== assign
% percentage
^ bitwise
& reference
* star
+ add
- sub
\ backslash
/ slash

Input.C:

```
#include "stdio.h"  
#include "conio.h"  
void main()  
{  
int a=10,b,c;  
a=b*c;  
getch();  
}
```

Sample Input & Output:

Enter the File Name: Input.C

Line: 1

: Preprocessor
include : keyword
< : lesser
stdio.h : keyword
> : greater

Line: 2

: Preprocessor
include : keyword
< : lesser
conio.h : keyword
> : greater

Line: 3

void : keyword

```
main : keyword  
( : openpara  
) : closepara
```

```
Line: 4  
{ : openbrace
```

```
Line: 5  
int : keyword  
a : identifier  
= : equal  
10 : constant  
, : identifier  
b : identifier  
, : identifier  
c : identifier  
; : semicolon
```

```
Line: 6  
a : identifier  
= : equal  
b : identifier  
* : star  
c : identifier  
; : semicolon
```

```
Line: 8  
} : closebrace
```

Result:

The above C program was successfully executed and verified.

Ex. No: 4

Date:

Construction of NFA from Regular Expression

Aim:

To write a C program to construct a Non Deterministic Finite Automata (NFA) from Regular Expression.

Algorithm:

1. Start the Program.
2. Enter the regular expression R over alphabet E.
3. Decompose the regular expression R into its primitive components
4. For each component construct finite automata.
5. To construct components for the basic regular expression way that corresponding to that way compound regular expression.
6. Stop the Program.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
#include<graphics.h>
#include<math.h>
#include<process.h>
int minx=1000,miny=0;
void star(int *x1,int *y1,int *x2,int *y2)
{
    char pr[10];
    ellipse(*x1+(*x2-*x1)/2,*y2-10,0,180,(*x2-*x1)/2,70);
    outtextxy(*x1-2,*y2-17,"v");
    line(*x2+10,*y2,*x2+30,*y2);
    outtextxy(*x1-15,*y1-3,">");
    circle(*x1-40,*y1,10);
    circle(*x1-80,*y1,10);
    line(*x1-30,*y2,*x1-10,*y2);
    outtextxy(*x2+25,*y2-3,">");
    sprintf(pr,"%c",238);
    outtextxy(*x2+15,*y2-9,pr);
    outtextxy(*x1-25,*y1-9,pr);
    outtextxy((*x2-*x1)/2+*x1,*y1-30,pr);
```



```

outtextxy((*x2-*x1)/2+*x1,*y1+30,pr);
ellipse(*x1+(*x2-*x1)/2,*y2+10,180,360,(*x2-*x1)/2+40,70);
outtextxy(*x2+37,*y2+14,"^");
if(*x1-40<minx)minx=*x1-40;
miny=*y1;
}
void star1(int *x1,int *y1,int *x2,int *y2)
{
char pr[10];
ellipse(*x1+(*x2-*x1)/2+15,*y2-10,0,180,(*x2-*x1)/2+15,70);
outtextxy(*x1-2,*y2-17,"v");
line(*x2+40,*y2,*x2+60,*y2);
outtextxy(*x1-15,*y1-3,">");
circle(*x1-40,*y1,10);
line(*x1-30,*y2,*x1-10,*y2);
outtextxy(*x2+25,*y2-3,">");
sprintf(pr,"%c",238);
outtextxy(*x2+15,*y2-9,pr);
outtextxy(*x1-25,*y1-9,pr);
outtextxy((*x2-*x1)/2+*x1,*y1-30,pr);
outtextxy((*x2-*x1)/2+*x1,*y1+30,pr);
ellipse(*x1+(*x2-*x1)/2+15,*y2+10,180,360,(*x2-*x1)/2+50,70);
outtextxy(*x2+62,*y2+13,"^");
if(*x1-40<minx)minx=*x1-40;
miny=*y1;
}
void basis(int *x1,int *y1,char x)
{
char pr[5];
circle(*x1,*y1,10);
line(*x1+30,*y1,*x1+10,*y1);
sprintf(pr,"%c",x);
outtextxy(*x1+20,*y1-10,pr);
outtextxy(*x1+23,*y1-3,">");
circle(*x1+40,*y1,10);
if(*x1<minx)minx=*x1;
miny=*y1;
}
void slash(int *x1,int *y1,int *x2,int *y2,int *x3,int *y3,int *x4,int *y4)
{
char pr[10];
int c1,c2;
c1=*x1;
if(*x3>c1)c1=*x3;
c2=*x2;
if(*x4>c2)c2=*x4;
line(*x1-10,*y1,c1-40,(*y3-*y1)/2+*y1-10);

```

```

outtextxy(*x1-15,*y1-3,">");
outtextxy(*x3-15,*y4-3,">");
circle(c1-40,(*y4-*y2)/2+*y2,10);
sprintf(pr,"%c",238);
outtextxy(c1-40,(*y4-*y2)/2+*y2+25,pr);
outtextxy(c1-40,(*y4-*y2)/2+*y2-25,pr);
line(*x2+10,*y2,c2+40,(*y4-*y2)/2+*y2-10);
line(*x3-10,*y3,c1-40,(*y3-*y1)/2+*y2+10);
circle(c2+40,(*y4-*y2)/2+*y2,10);
outtextxy(c2+40,(*y4-*y2)/2+*y2-25,pr);
outtextxy(c2-40,(*y4-*y2)/2+*y2+25,pr);
outtextxy(c2+35,(*y4-*y2)/2+*y2-15,"^");
outtextxy(c1+35,(*y4-*y2)/2+*y2+10,"^");
line(*x4+10,*y2,c2+40,(*y4-*y2)/2+*y2+10);
minx=c1-40;
miny=(*y4-*y2)/2+*y2;
}
void main()
{
int d=0,l,x1=200,y1=200,len,par=0,op[10];
int cx1=200,cy1=200,cx2,cy2,cx3,cy3,cx4,cy4;
char str[20];
int gd=DETECT,gm;
int stx[20],endx[20],sty[20],endy[20];
int pos=0,i=0;
clrscr();
initgraph(&gd,&gm,"c:\\dosapp\\tcplus\\bgi");
printf("\n enter the regular expression:");
scanf("%s",str);
len=(strlen(str));
while(i<len)
{
if(isalpha(str[i]))
{
if(str[i+1]=='*')x1=x1+40;
basis(&x1,&y1,str[i]);
stx[pos]=x1;
endx[pos]=x1+40;
sty[pos]=y1;
endy[pos]=y1;
x1=x1+40;
pos++;
}
if(str[i]=='*')
{
star(&stx[pos-1],&sty[pos-1],&endx[pos-1],&endy[pos-1]);
stx[pos-1]=stx[pos-1]-40;

```

```

endx[pos-1]=endx[pos-1]+40;
x1=x1+40;
}
if(str[i]=='(')
{
int s;
s=i;
while(str[s]!='')s++;
if((str[s+1]=='*')&&(pos!=0))x1=x1+40;
op[par]=pos;
par++;
}
if(str[i]==')')
{
cx2=endx[pos-1];
cy2=endy[pos-1];
l=op[par-1];
cx1=stx[1];
cx2=sty[1];
par--;
if(str[i+1]=='*')
{
i++;
star1(&cx1,&cy1,&cx2,&cy2);
cx1=cx1-40;
cx2=cx2+40;
stx[1]=stx[1]-40;
endx[pos-1]=endx[pos-1]+40;
x1=x1+40;
}
if(d==1)
{
slash(&cx3,&cy3,&cx4,&cy4,&cx1,&cy1,&cx2,&cy2);
if(cx4>cx2)x1=cx4+40;
else x1=cx2+40;
y1=(y1-cy4)/2.0+cy4;
d=0;
}
}
if(str[i]=='/')
{
cx2=endx[pos-1];
cy2=endy[pos-1];
x1=200;
y1=y1+100;

```

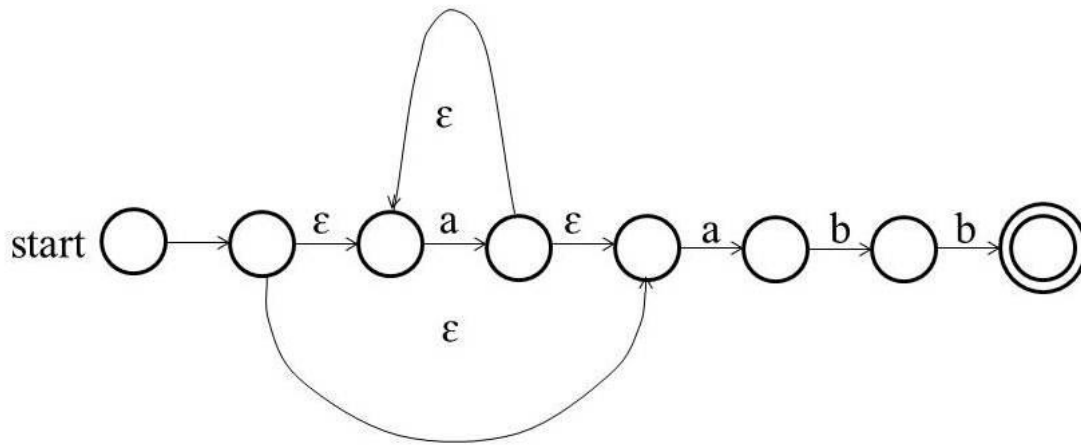
```

if(str[i+1]=='(')
{
d=1;
cx3=cx1;
cy3=cy1;
cx4=cx2;
cy4=cy2;
}
if(isalpha(str[i+1]))
{
i++;
basis(&x1,&y1,str[i]);
stx[pos]=x1;
endx[pos]=x1+40;
sty[pos]=y1;
endy[pos]=y1;
if(str[i+1]=='*')
{
i++;
star(&stx[pos],&sty[pos],&endx[pos],&endy[pos]);
stx[pos]=stx[pos]-40;
endx[pos]=endx[pos]+40;
}
slash(&cx1,&cy1,&cx2,&cy2,&stx[pos],&sty[pos],&endx[pos],&endy[pos]);
if(cx2>endx[pos])x1=cx2+40;
else x1=endx[pos]+40;
y1=(y1-cy2)/2.0+cy2;
cx1=cx1-40;
cy1=(sty[pos]-cy1)/2.0+cy1;
cx2=cx2+40;
cy2=(endy[pos]-cy2)/2.0+cy2;
l=op[par-1];
stx[1]=cx1;
sty[1]=cy1;
endx[pos]=cx2;
endy[pos]=cy2;
pos++;
}
}
i++;
}
circle(x1,y1,13);
line(minx-30,miny,minx-10,miny);
outtextxy(minx-100,miny-10,"start");
outtextxy(minx-15,miny-3,">");

```

```
getch();  
closegraph();  
}
```

Sample Input & Output:



Result:

The above C program was successfully executed and verified.

Ex.No: 5

Date:

Construction of DFA from NFA

Aim:

To write a C program to construct a DFA from the given NFA.

Algorithm:

1. Start the program.
2. Accept the number of state A and B.
3. Find the E-closure for node and name if as A.
4. Find $v(a,a)$ and (a,b) and find a state.
5. Check whether a number new state is obtained.
6. Display all the state corresponding A and B.
7. Stop the program.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<process.h>
typedef struct
{
int num[10],top;
}
stack;
stack s;
int mark[16][31],e_close[16][31],n,st=0;
char data[15][15];
void push(int a)
{
s.num[s.top]=a;
s.top=s.top+1;
}
int pop()
{
int a;
if(s.top==0)
return(-1);
s.top=s.top-1;
a=s.num[s.top];
```

```

return(a);
}
void epi_close(int s1,int s2,int c)
{
int i,k,f;
for(i=1;i<=n;i++)
{
if(data[s2][i]=='e')
{
f=0;
for(k=1;k<=c;k++)
if(e_close[s1][k]==i)
f=1;
if(f==0)
{
c++;
e_close[s1][c]=i;
push(i);
}
}
}
while(s.top!=0) epi_close(s1,pop(),c);
}
int move(int sta,char c)
{
int i;
for(i=1;i<=n;i++)
{
if(data[sta][i]==c)
return(i);
}
return(0);
}
void e_union(int m,int n)
{
int i=0,j,t;
for(j=1;mark[m][i]!=-1;j++)
{
while((mark[m][i]!=e_close[n][j])&&(mark[m][i]!=-1))
i++;
if(mark[m][i]==-1)mark[m][i]=e_close[n][j];
}
}
void main()
{
int i,j,k,Lo,m,p,q,t,f;
clrscr();

```

```

printf("\n enter the NFA state table entries:");
scanf("%d",&n);
printf("\n");
for(i=0;i<=n;i++)
printf("%d",i);
printf("\n");
for(i=0;i<=n;i++)
printf("-----");
printf("\n");
for(i=1;i<=n;i++)
{
printf("%d|",i);
fflush(stdin);
for(j=1;j<=n;j++)
scanf("%c",&data[i][j]);
}
for(i=1;i<=15;i++)
for(j=1;j<=30;j++)
{
e_close[i][j]=-1;
mark[i][j]=-1;
}
for(i=1;i<=n;i++)
{
e_close[i][1]=i;
s.top=0;
epi_close(i,i,1);
}
for(i=1;i<=n;i++)
{
for(j=1;e_close[i][j]!=-1;j++)
for(k=2;e_close[i][k]!=-1;k++)
if(e_close[i][k-1]>e_close[i][k])
{
t=e_close[i][k-1];
e_close[i][k-1]=e_close[i][k];
e_close[i][k]=t;
}
}
printf("\n the epsilon closures are:");
for(i=1;i<=n;i++)
{
printf("\n E(%d)={",i);
for(j=1;e_close[i][j]!=-1;j++)
printf("%d",e_close[i][j]);
printf("}");
}

```



```

j=1;
while(e_close[1][j]!=-1)
{
mark[1][j]=e_close[1][j];
j++;
}
st=1;
printf("\n DFA Table is:");
printf("\n          a          b      ");
printf("\n-----");
for(i=1;i<=st;i++)
{
printf("\n{ ");
for(j=1;mark[i][j]!=-1;j++)
printf("%d",mark[i][j]);
printf("}");
while(j<7)
{
printf(" ");
j++;
}
for(Lo=1;Lo<=2;Lo++)
{
for(j=1;mark[i][j]!=-1;j++)
{
if(Lo==1)
t=move(mark[i][j],'a');
if(Lo==2)
t=move(mark[i][j],'b');
if(t!=0)
e_union(st+1,t);
}
for(p=1;mark[st+1][p]!=-1;p++)
for(q=2;mark[st+1][q]!=-1;q++)
{
if(mark[st+1][q-1]>mark[st+1][q])
{
t=mark[st+1][q];
mark[st+1][q]=mark[st+1][q-1];
mark[st+1][q-1]=t;
}
}
}
f=1;
for(p=1;p<=st;p++)
{
j=1;

```

```

while((mark[st+1][j]==mark[p][j])&&(mark[st+1][j]!=-1))
j++;
if(mark[st+1][j]==-1 && mark[p][j]==-1)
f=0;
}
if(mark[st+1][1]==-1)
f=0;
printf("\t{ ");
for(j=1;mark[st+1][j]!=-1;j++)
{
printf("%d",mark[st+1][j]);
}
printf("}\t");
if(Lo==1)
printf(" ");
if(f==1)
st++;
if(f==0)
{
for(p=1;p<=30;p++)
mark[st+1][p]=-1;
}
}
}
getch();
}

```

Sample Input & Output:

Enter the NFA state table entries: 11

(Note: Instead of '-' symbol use blank spaces in the output window)

0 1 2 3 4 5 6 7 8 9 10 11

```
-----  
1 - e - - - - e - - -  
2 - - e - e - - - - -  
3 - - - a - - - - - -  
4 - - - - - e - - - -  
5 - - - - - b - - - -  
6 - - - - - e - - - -  
7 - e - - - - e - - -  
8 - - - - - - - e - -  
9 - - - - - - - e - -  
10 - - - - - - - - e  
11 - - - - - - - - -
```

The Epsilon Closures Are:

$E(1) = \{12358\}$

$E(2) = \{235\}$

$E(3) = \{3\}$

$E(4) = \{234578\}$

$E(5) = \{5\}$

$E(6) = \{235678\}$

$E(7) = \{23578\}$

$E(8) = \{8\}$

$E(9) = \{9\}$

$E(10) = \{10\}$

$E(11) = \{11\}$

DFA Table is:

	a	b
{12358}	{2345789}	{235678}
{2345789}	{2345789}	{23567810}
{235678}	{2345789}	{235678}
{23567810}	{2345789}	{23567811}
{23567811}	{2345789}	{235678}

Result:

The above C program was successfully executed and verified.

Ex.No: 6

Date:

Implementation of Shift Reduce Parsing Algorithm

Aim:

To write a C program to implement the shift-reduce parsing algorithm.

Algorithm:

Grammar:

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow E / E$

$E \rightarrow a/b$

Method:

<u>Stack</u>	<u>Input Symbol</u>	<u>Action</u>
\$	id1*id2\$	shift
\$id1	*id2 \$	shift *
\$*	id2\$	shift id2
\$id2	\$	shift
\$	\$	accept

Shift: Shifts the next input symbol onto the stack.

Reduce: Right end of the string to be reduced must be at the top of the stack.

Accept: Announce successful completion of parsing.

Error: Discovers a syntax error and call an error recovery routine.

Program:

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
void main()
{
clrscr();
printf("\n\n\t Shift Reduce Parser\n");
printf("\n\t***** ***** *****");
printf("\n Grammar\n\n");
printf("E->E+E\nE->E/E\n");
printf("E->E*E\nE->a/b");
printf("\n Enter the Input Symbol:\t");
gets(ip_sym);
printf("\n\n\t Stack Implementation Table");
printf("\n Stack\t\t Input Symbol\t\t Action");
printf("\n $ \t\t %s \t\t --",ip_sym);
strcpy(act,"shift");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
len=strlen(ip_sym);
for(i=0;i<=len-1;i++)
{
stack[st_ptr]=ip_sym[ip_ptr];
stack[st_ptr+1]='\0';
ip_sym[ip_ptr]=' ';
ip_ptr++;
printf("\n$%s\t\t%s\t\t%s",stack,ip_sym,act);
strcpy(act,"shift");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
check();
st_ptr++;
}
st_ptr++;
check();
getch();
}
```

```

void check()
{
int flag=0;
temp2[0]=stack[st_ptr];
temp[1]='\0';
if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))
{
stack[st_ptr]='E';
if(!strcmpi(temp2,"a"))
printf("\n%s\t\t%s\t\tE->a",stack,ip_sym);
else
printf("\n%s\t\t%s\t\tE->a",stack,ip_sym);
flag=1;
}
if((!strcmpi(temp2,"+"))||(strcmpi(temp2,"*"))||(!strcmpi(temp2,"/")))
{
flag=1;
}
if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E/E"))||(!strcmpi(stack,"E*E")))
{
strcpy(stack,"E");
st_ptr=0;
if(!strcmpi(stack,"E+E"))
printf("\n%s\t\t%s\t\tE->E+E",stack,ip_sym);
else
if(!strcmpi(stack,"E/E"))
printf("\n%s\t\t%s\t\tE->E/E",stack,ip_sym);
else
printf("\n%s\t\t%s\t\tE->E*E",stack,ip_sym);
flag=1;
}
if(!strcmpi(stack,"E")&&ip_ptr==len)
{
printf("\n%s\t\t%s\t\tAccept",ip_sym);
getch();
exit(0);
}
if(flag==0)
{
printf("\n %s \t\t %s \t\t Reject",stack,ip_sym);
}
return;
}

```

Sample Input & Output:

Shift Reduce Parser

Grammar

E->E+E

E->E/E

E->E*E

E->a/b

Enter the input symbol: if(a*b)

Stack Implementation Table

Stack	Input Symbol	Action
\$	if(a*b)\$	--
\$i	f(a*b)\$	shift i
\$if	(a*b)\$	shift f
\$if(a*b)\$	shift (
\$if(a	*b)\$	shift a
\$if(E	*b)\$	E->a
\$if(E*	b)\$	shift *
if(E*	b)	reject

Press any key to continue...

Result:

The above C program was successfully executed and verified.

Date:

Aim:

Algorithm:

Output: Sequence of shift reduce step1

- 1- Let the input string to be initially the stack contains, when the reduce action takes place we have to reach create parent child relationship.
- 2- See IP to pointer to the first symbol of input string and repeat forever if only \$ is on the input accept and break else begin.
- 3- Let 'd' be the top most terminal on the stack and 'b' be current input IF($a < b$) or $a = b$ then Begin push 'b' onto the stack.
- 4- Advance Input to the stack to the next Input symbol
end;
else if($a > b$)
- 5- Repeat pop the stack until the top most terminal is related by $<$ to the terminal most recently popped else error value routine
end;

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
char q[9][9]={
{'>','>','<','<','<','<','>','<','>'},
{'>','>','<','<','<','<','>','<','>'},
{'>','>','>','>','<','<','>','<','>'},
{'>','>','>','>','<','<','>','<','>'},
{'>','>','<','<','<','<','>','<','>'}
```



```

}
intrel(char a,char b,char d)
{
if(isalpha(a)!=0)
a='a';
if(isalpha(b)!=0)
b='a';
if(q[find(a)][find(b)]==d)
return 1;
else
return 0;
}
void main()
{
char s[100];
int i=-1;
clrscr();
printf("\n\t Operator Preceding Parser\n");
printf("\n Enter the Arithmetic Expression End with $..");
gets(s);
push('$');
while(i)
{
if((s[p]=='$')&&(st[top]=='$'))
{
printf("\n\nAccepted");
break;
}
else if(rel(st[top],s[p], '<') || rel(st[top],s[p], '='))
{
display(s[p]);
push(s[p]);
p++;
}
else if(rel(st[top],s[p], '>'))
{
do
{
r++;
qs[r]=pop();
display1(qs[r]);
}
while(!rel(st[top],qs[r], '<'));
}
}
getch();
}

```

Sample Input & Output:

Enter the Arithmetic Expression End with \$: a-(b*c)^d\$

Shift a
Reduce E->a
Shift -
Shift (
Shift b
Reduce E->b
Shift *
Shift c
Reduce E->c
Reduce E->E*E
Shift)
Reduce E->(E)
Shift ^
Shift d
Reduce E->d
Reduce E->E^E
Reduce E->E-E
Accepted

Result:

The above C program was successfully executed and verified.

Ex.No: 8

Date:

Implementation of Recursive Descent Parser

Aim:

To write a C program to implement Recursive Descent Parser.

Algorithm:

Input: Context Free Grammar without last recursion and an input string from the grammar.

Output: Sequence of productions rules used to derive the sentence.

Method:

Consider the grammar

$E \rightarrow TE$

$E' \rightarrow +TE'/e$

$T \rightarrow FT$

$T \rightarrow *FT/e$

$F \rightarrow (E)/Id$

To recursive decent parser for the above grammar is given below

Procedure:

Begin

T()

E_prime();

print E-> TE'

end

procedureeprime():

if ip_sym == '+' then

begin

advance();

T();

eprime();

prime E'->TE'

end

else

print E'->e

procedure T();

begin

e();

```

Tprime();
print T->FT';
end;

procedure Tprime();
if ip_sym='*' then
begin
advance();
F();
Tprime()
print T'->T*FT'
end
else print T'->e

procedure F()
if ip_sym=id then
begin
advance();
print->id
end
else
Error();
end;
else
Error();

```

Program:

```

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
char ip_sym[15], ip_ptr=0;
void e_prime();
void t();
void e();
void t_prime();
void f();
void advance();
void e()
{
printf("\n\t\tE'----->TE");
t();
e_prime();
}
void e_prime()
{

```

```

if(ip_sym[ip_ptr]=='+')
{
printf("\n\t\tE'----->+TE");
advance();
t();
e_prime();
}
else
printf("\n\t\tE'----->e");
}
void t()
{
printf("\n\t\tT'----->FT"); f();
t_prime();
}
void t_prime()
{
if(ip_sym[ip_ptr]=='*')
{
printf("\n\t\tT'----->*FT"); advance();
f();
t_prime();
}
else
{
printf("\n\t\tT'----->e");
}
}
void f()
{
if((ip_sym[ip_ptr]=='i')||(ip_sym[ip_ptr]=='j'))
{
printf("\n\t\tF'----->i"); advance();
}
else
{
if(ip_sym[ip_ptr]=='(')
{
advance();
e();
if(ip_sym[ip_ptr]==')')
{
advance();
printf("\n\t\tF'----->(E)");
}
}
}
}

```

```

else
{
printf("\n\tSyntax Error");
getch();
exit(1);
}
}
}
}
void advance()
{
ip_ptr++;
}
void main()
{
int i;
clrscr();
printf("\n\tGRAMMER WITHOUT RECURSION");
printf("\n\tE----->TE\n\tE'/e\r\tT----->FT");
printf("\n\tT----->*FT/e\n\tF----->(E)/id");
printf("\n\tEnter the Input Symbol: ");
gets(ip_sym);
printf("\n\tSequence of Production Rules");
e();
getch();
}

```

Sample Input & Output:

GRAMMER WITHOUT RECURSION

E----->TE'

T----->FT

T----->*FT/e

F----->(E)/id

Enter the Input Symbol: T

Sequence of Production Rules

E'----->TE'

T'----->FT'

T'----->e

E'----->e'

Result:

The above C program was successfully executed and verified.

Ex.No: 9

Date:

Implementation of Code Optimization Techniques

Aim:

To write a C program to implement Code Optimization Techniques.

Algorithm:

Input: Set of 'L' values with corresponding 'R' values.

Output: Intermediate code & Optimized code after eliminating common expressions.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct op
{
    char l;
    char r[20];
}
op[10],pr[10];
void main()
{
    int a,i,k,j,n,z=0,m,q;
    char *p,*l;
    char temp,t;
    char *tem;
    clrscr();
    printf("Enter the Number of Values:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("left: ");
        op[i].l=getche();
        printf("\tright: ");
        scanf("%s",op[i].r);
    }
    printf("Intermediate Code\n") ;
    for(i=0;i<n;i++)
    {
        printf("%c=",op[i].l);
        printf("%s\n",op[i].r);
    }
}
```

```

for(i=0;i<n-1;i++)
{
temp=op[i].l;
for(j=0;j<n;j++)
{
p=strchr(op[j].r,temp);
if(p)
{
pr[z].l=op[i].l;
strcpy(pr[z].r,op[i].r);
z++;
}
}
}
pr[z].l=op[n-1].l;
strcpy(pr[z].r,op[n-1].r);
z++;
printf("\nAfter Dead Code Elimination\n");
for(k=0;k<z;k++)
{
printf("%c\t=",pr[k].l);
printf("%s\n",pr[k].r);
}
for(m=0;m<z;m++)
{
tem=pr[m].r;
for(j=m+1;j<z;j++)
{
p=strstr(tem,pr[j].r);
if(p)
{
t=pr[j].l;
pr[j].l=pr[m].l;
for(i=0;i<z;i++)
{
l=strchr(pr[i].r,t) ;
if(l)
{
a=l-pr[i].r;
printf("pos: %d",a);
pr[i].r[a]=pr[m].l;
}
}
}
}
}
}
}

```

```

printf("Eliminate Common Expression\n");
for(i=0;i<z;i++)
{
printf("%c\t=",pr[i].l);
printf("%s\n",pr[i].r);
}
for(i=0;i<z;i++)
{
for(j=i+1;j<z;j++)
{
q=strcmp(pr[i].r,pr[j].r);
if((pr[i].l==pr[j].l)&&!q)
{
pr[i].l='\0';
strcpy(pr[i].r,"\0");
}
}
}
printf("Optimized Code\n");
for(i=0;i<z;i++)
{
if(pr[i].l!='\0')
{
printf("%c=",pr[i].l);
printf("%s\n",pr[i].r);
}
}
getch();
}

```

Sample Input & Output:

Enter the Number of Values: 5

Left: a right: 9

Left: b right: c+d

Left: e right: c+d

Left: f right: b+e

Left: r right: f

Intermediate Code

a=9

b=c+d

e=c+d

f=b+e

r=:f

After Dead Code Elimination

b =c+d

e =c+d

f =b+e

r =:f

Eliminate Common Expression

b =c+d

b =c+d

f =b+b

r =:f

Optimized Code

b=c+d

f=b+b

r=:f

Result:

The above C program was successfully executed and verified.

Ex.No: 10

Date:

Implementation of Code Generator

Aim:

To write a C program to implement Simple Code Generator.

Algorithm:

Input: Set of three address code sequence.

Output: Assembly code sequence for three address codes (opd1=opd2, op, opd3).

Method:

- 1- Start
- 2- Get address code sequence.
- 3- Determine current location of 3 using address (for 1st operand).
- 4- If current location not already exist generate move (B,O).
- 5- Update address of A(for 2nd operand).
- 6- If current value of B and () is null,exist.
- 7- If they generate operator () A,3 ADPR.
- 8- Store the move instruction in memory
- 9- Stop.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<graphics.h>
typedef struct
{
    char var[10];
    int alive;
}
regist;
regist preg[10];
void substring(char exp[],int st,int end)
{
    int i,j=0;
    char dup[10]="";
    for(i=st;i<end;i++)
        dup[j++]=exp[i];
    dup[j]='0';
```

```

strcpy(exp,dup);
}
int getregister(char var[])
{
int i;
for(i=0;i<10;i++)
{
if(preg[i].alive==0)
{
strcpy(preg[i].var,var);
break;
}
}
return(i);
}
void getvar(char exp[],char v[])
{
int i,j=0;
char var[10]="";
for(i=0;exp[i]!='\0';i++)
if(isalpha(exp[i]))
var[j++]=exp[i];
else
break;
strcpy(v,var);
}
void main()
{
char basic[10][10],var[10][10],fstr[10],op;
int i,j,k,reg,vc,flag=0;
clrscr();
printf("\nEnter the Three Address Code:\n");
for(i=0;;i++)
{
gets(basic[i]);
if(strcmp(basic[i],"exit")==0)
break;
}
printf("\nThe Equivalent Assembly Code is:\n");
for(j=0;j<i;j++)
{
getvar(basic[j],var[vc++]);
strcpy(fstr,var[vc-1]);
substring(basic[j],strlen(var[vc-1])+1,strlen(basic[j]));
getvar(basic[j],var[vc++]);
reg=getregister(var[vc-1]);

```

```

if(preg[reg].alive==0)
{
printf("\nMov R%d,%s",reg,var[vc-1]);
preg[reg].alive=1;
}
op=basic[j][strlen(var[vc-1])];
substring(basic[j],strlen(var[vc-1])+1,strlen(basic[j]));
getvar(basic[j],var[vc++]);
switch(op)
{
case '+': printf("\nAdd"); break;
case '-': printf("\nSub"); break;
case '*': printf("\nMul"); break;
case '/': printf("\nDiv"); break;
}
flag=1;
for(k=0;k<=reg;k++)
{
if(strcmp(preg[k].var,var[vc-1])==0)
{
printf("R%d, R%d",k,reg);
preg[k].alive=0;
flag=0;
break;
}
}
if(flag)
{
printf(" %s,R%d",var[vc-1],reg);
printf("\nMov %s,R%d",fstr,reg);
}
strcpy(preg[reg].var,var[vc-3]);
getch();
}
}

```

Sample Input & Output:

Enter the Three Address Code:

a=b+c

c=a*c

exit

The Equivalent Assembly Code is:

Mov R0,b

Add c,R0

Mov a,R0

Mov R1,a

Mul c,R1

Mov c,R1

Result:

The above C program was successfully executed and verified.

Ex.No: 11 (A)

Date:

To Find IP Address of The Local Host

Aim:

To write a Java program to find the IP Address of a Local Host.

Class:

InetAddress

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. There are no public constructors in the InetAddress class.

However, InetAddress has three static methods that return suitably initialized InetAddress objects. They are:

`static InetAddress getLocalHost() :`

The getLocalHost() method simply returns the InetAddress object that represents the local host. If these methods are unable to resolve the host name, they throw an UnknownHostException.

`static InetAddress getByName(String hostName) :`

The getByName() method returns an InetAddress for a host name passed to it.

`static InetAddress[] getAllByName(String hostName) :`

The getAllByName() factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can't resolve the name to at least one address.

Program:

```
import java.net.*;
class ex11a
{
public static void main(String args[])throws UnknownHostException
{
InetAddress Ia=InetAddress.getLocalHost();
System.out.println(Ia);
}
}
```

Sample Input& Output:

```
D:\Program Files\Java\jdk1.6.0_20\bin>javac ex11a.java  
D:\Program Files\Java\jdk1.6.0_20\bin> java ex11a
```

Sys-166/192.168.1.166

Result:

Thus the above Java Program was successfully executed and verified.

Ex.No:11(B)

Date:

To Find IP Address of The Remote Host

Aim:

To write a Java program to find the IP Address of a Remote Host.

Class:

InetAddress

The InetAddress class is used to encapsulate both the numerical IP address and the domain name for that address. There are no public constructors in the InetAddress class.

However, InetAddress has three static methods that return suitably initialized InetAddress objects. They are:

static InetAddress getLocalHost() :

The getLocalHost() method simply returns the InetAddress object that represents the local host. If these methods are unable to resolve the host name, they throw an UnknownHostException.

static InetAddress getByName(String hostName) :

The getByName() method returns an InetAddress for a host name passed to it.

static InetAddress[] getAllByName(String hostName) :

The getAllByName() factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can't resolve the name to at least one address.

Program:

```
import java.net.*;
import java.lang.*;
class ex11b
{
public static void main(String args[])throws UnknownHostException
{
InetAddress add=InetAddress.getByName(args[0]);
System.out.println("The InetAddress:"+add);
}
}
```

Sample Input & Output:

```
D:\Program Files\Java\jdk1.6.0_20\bin>javac ex11b.java  
D:\Program Files\Java\jdk1.6.0_20\bin> java ex11b sys-165
```

The Inet Address: sys-165/199.168.1.165

Result:

Thus the above Java Program was successfully executed and verified.

Ex.No:12

Date:

Implementation of Echo Server and Client using TCP Sockets

Aim:

To implement echo server and client in java using TCP sockets.

Algorithm:

Server:

1. Create a server socket.
2. Wait for client to be connected.
3. Read text from the client
4. Echo the text back to the client.
5. Repeat steps 4-5 until 'bye' or 'null' is read.
6. Close the I/O streams
7. Close the server socket
8. Stop

Client:

1. Create a socket and establish connection with the server
2. Get input from user.
3. If equal to bye or null, then go to step 7.
4. Send text to the server.
5. Display the text echoed by the server
6. Repeat steps 2-4
7. Close the I/O streams
8. Close the client socket
9. Stop

Program:

TCP Echo Server - tcpechoserver.java

```
import java.net.*;
import java.io.*;
public class tcpechoserver
{
    public static void main(String[] arg) throws IOException
    {
        ServerSocket sock = null;
        BufferedReader fromClient = null;
        OutputStreamWriter toClient = null;
        Socket client = null;
```

```

try
{
sock = new ServerSocket(4000);
System.out.println("Server Ready");
client = sock.accept();
System.out.println("Client Connected");
fromClient = new BufferedReader(new
InputStreamReader(client.getInputStream()));
toClient = new OutputStreamWriter(client.getOutputStream());
String line;
while (true)
{
line = fromClient.readLine();
if ((line == null) || line.equals("bye"))
break;
System.out.println ("Client [ " + line + " ]");
toClient.write("Server [ "+ line +" ]\n");
toClient.flush();
}
fromClient.close();
toClient.close();
client.close();
sock.close();
System.out.println("Client Disconnected");
}
catch (IOException ioe)
{
System.err.println(ioe);
}
}
}

```

TCP Echo Client - tcpechoclient.java

```
import java.net.*;
import java.io.*;
public class tcpechoclient
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader fromServer = null;
        BufferedReader fromUser = null;
        PrintWriter toServer = null;
        Socket sock = null;
        try
        {
            if (args.length == 0)
                sock = new Socket(InetAddress.getLocalHost(),4000);
            else
                sock = new Socket(InetAddress.getByName(args[0]),4000);
            fromServer = new BufferedReader(new InputStreamReader(sock.getInputStream()));
            fromUser = new BufferedReader(new InputStreamReader(System.in));
            toServer = new PrintWriter(sock.getOutputStream(), true);
            String Usrmsg, Srvmsg;
            System.out.println("Type \"bye\" to quit");
            while (true)
            {
                System.out.print("Enter msg to server : ");
                Usrmsg = fromUser.readLine();
                if(Usrmsg==null || Usrmsg.equals("bye"))
                {
                    toServer.println("bye");
                    break;
                }
                else
                {
                    toServer.println(Usrmsg);
                    Srvmsg = fromServer.readLine();
                    System.out.println(Srvmsg);
                }
            }
            fromUser.close();
            fromServer.close();
            toServer.close();
            sock.close();
        }
        catch (IOException ioe)
        {
            System.err.println(ioe);
        }
    }
}
```

Sample Input & Output:**Server:**

```
$ javac tcpechoserver.java
$ java tcpechoserver
Server Ready
Client Connected
Client [ hello ]
Client [ how are you ]
Client [ i am fine ]
Client [ ok ]
Client Disconnected
```

Client:

```
$ javac tcpechoclient.java
$ java tcpechoclient
Type "bye" to quit
Enter msg to server : hello
Server [ hello ]
Enter msg to server : how are you
Server [ how are you ]
Enter msg to server : i am fine
Server [ i am fine ]
Enter msg to server : ok
Server [ ok ]
Enter msg to server : bye
```

Result:

Thus data from client to server is echoed back to the client to check reliability/noise level of the channel.

Ex. No: 13

Date:

Implementation of Echo Server and Client using UDP Sockets

Aim:

To implement date server and client in java using TCP sockets.

Algorithm:

Server:

1. Create an array of hosts and its ip address in another array
2. Create a datagram socket and bind it to a port
3. Create a datagram packet to receive client request
4. Read the domain name from client to be resolved
5. Lookup the host array for the domain name
6. If found then retrieve corresponding address
7. Create a datagram packet and send ip address to client
8. Repeat steps 3-7 to resolve further requests from clients
9. Close the server socket
10. Stop

Client:

1. Create a datagram socket
2. Get domain name from user
3. Create a datagram packet and send domain name to the server
4. Create a datagram packet to receive server message
5. Read server's response
6. If ip address then display it else display "Domain does not exist"
7. Close the client socket
8. Stop

Program:

UDP DNS Server -- udpdnsserver.java

```
import java.io.*;
import java.net.*;
public class udpdnsserver
{
    private static int indexOf(String[] array, String str)
    {
        str = str.trim();
        for (int i=0; i < array.length; i++)
        {
            if (array[i].equals(str))
                return i;
        }
    }
}
```

```

return -1;
}
public static void main(String arg[])throws IOException
{
String[] hosts = {"yahoo.com", "gmail.com", "google.com", "facebook.com"};
String[] ip = {"68.180.206.184", "209.85.148.19", "80.168.92.140", "69.63.189.16"};
System.out.println("Press Ctrl + C to Quit");
while (true)
{
DatagramSocket serversocket=new DatagramSocket(1362);
byte[] senddata = new byte[1021];
byte[] receivedata = new byte[1021];
DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);
serversocket.receive(recvpack);
String sen = new String(recvpack.getData());
InetAddress ipaddress = recvpack.getAddress();
int port = recvpack.getPort();
String capsent;
System.out.println("Request for host " + sen);
if(indexOf (hosts, sen) != -1)
capsent = ip[indexOf (hosts, sen)];
else
capsent = "Host Not Found";
senddata = capsent.getBytes();
DatagramPacket pack = new DatagramPacket(senddata,
senddata.length,ipaddress,port);
serversocket.send(pack);
serversocket.close();
}
}
}

```

UDP DNS Client -- udpdnsclient.java

```

import java.io.*;
import java.net.*;
public class udpdnsclient
{
public static void main(String args[])throws IOException
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientsocket = new DatagramSocket();
InetAddress ipaddress;
if(args.length == 0)
ipaddress = InetAddress.getLocalHost();
else
ipaddress = InetAddress.getByName(args[0]);

```

```

byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
int portaddr = 1362;
System.out.print("Enter the hostname : ");
String sentence = br.readLine();
senddata = sentence.getBytes();
DatagramPacket pack = new DatagramPacket(senddata, senddata.length,
ipaddress,portaddr); clientsocket.send(pack);
DatagramPacket recvpack =new DatagramPacket(receivedata, receivedata.length);
clientsocket.receive(recvpack);
String modified = new String(recvpack.getData());
System.out.println("IP Address: " + modified);
clientsocket.close();
}
}

```

Sample Input & Output:

Server:

```

$ javac udpdnsserver.java
$ java udpdnsserver Press Ctrl + C to Quit

```

```

Request for host yahoo.com
Request for host google.com
Request for host youtube.com

```

Client:

```

$ javac udpdnsclient.java

$ java udpdnsclient
Enter the hostname : yahoo.com
IP Address: 68.180.206.184

$ java udpdnsclient
Enter the hostname : google.com
IP Address: 80.168.92.140

$ java udpdnsclient
Enter the hostname : youtube.com
IP Address: Host Not Found

```

Result:

Thus domain name requests by the client are resolved into their respective logical address using lookup method.

Ex.No: 14

Date:

Send and Receive Message between Client and Server using TCP

Aim:

To write a Java program to send and receive message between Client and Server using TCP.

Class:

ServerSocket:

This class implements server sockets. A server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.

Methods:

ServerSocket()	Creates an unbound server socket
ServerSocket(int port)	Creates a server socket, bound to specified port
accept()	Listens for a connection to be made to this socket and accepts it.
getInetAddress()	Returns the local address of this server socket.
getLocalPort()	Returns the port on which this socket is listening.
close()	Closes this socket

Program:

Client:

```
import java.net.*;
import java.io.*;
import java.lang.*;
public class ex14client
{
    public static void main(String args[])throws IOException
    {
        InetAddress a=InetAddress.getLocalHost();
        Socket soc=new Socket(a,8888);
```

```

try
{
    BufferedReader in=new BufferedReader(new
    InputStreamReader(soc.getInputStream()));
    PrintWriter out =new PrintWriter(new BufferedWriter(new
    OutputStreamWriter(soc.getOutputStream())),true);
    for(int i=0;i<5;i++)
    {
        out.println("lines#"+i);
        String str=in.readLine();
        System.out.println(str);
    }
    out.println("end");
}
finally
{
    System.out.println("closing");
    soc.close();
}
}
}

```

Server:

```

import java.io.*;
import java.net.*;
import java.lang.*;
public class ex14server
{
    public static final int PORT=8888;
    public static void main(String args[ ]) throws IOException
    {
        ServerSocket s=new ServerSocket(PORT);
        try
        {
            Socket soc=s.accept( );
            try
            {
                System.out.println("Connection accepted.....");
                BufferedReader in=new BufferedReader(new
                InputStreamReader(soc.getInputStream()));
                PrintWriter out =new PrintWriter(new BufferedWriter(new
                OutputStreamWriter(soc.getOutputStream())),true);

                while(true)
                {
                    String str=in.readLine();
                    if(str.equals("end"))

```

```

break;
System.out.println("echoing .."+str);
out.println(str);
}
}
finally
{
System.out.println("closing");
soc.close();
}
}
finally
{
s.close();
}
}
}

```

Sample Input & Output:

Client:

```

D:\Program Files\Java\jdk1.6.0_20\bin>javacex14client.java
D:\Program Files\Java\jdk1.6.0_20\bin> java ex14client

```

```

Connection accepted.....
'echoing ..lines#0
'echoing ..lines#1
'echoing ..lines#2
'echoing ..lines#3
'echoing ..lines#4
closing

```

Server:

```

D:\Program Files\Java\jdk1.6.0_20\bin>javacex14server.java
D:\Program Files\Java\jdk1.6.0_20\bin> java ex14server

```

```

lines#0
lines#1
lines#2
lines#3
lines#4
closing

```

Result:

Thus the above Java Program was successfully executed and verified.

Ex.No: 15

Date:

Send and Receive Message between Client and Server using UDP

Aim:

To write a Java program to send and receive message between Client and Server using UDP.

Class:

DatagramSocket:

This class represents a socket for sending and receiving datagram packets. A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order. User Datagram Protocol (UDP) broadcasts always enabled on a DatagramSocket. In order to receive broadcast packets a DatagramSocket should be bound to the wildcard address. In some implementations, broadcast packets may also be received when a DatagramSocket is bound to a more specific address.

DatagramPacket:

Datagram packets are used to implement a connectionless packet delivery service. Multiple packets sent from one machine to another might be routed differently, and might arrive in any order. Packet delivery is not guaranteed.

Methods:

DatagramSocket(int port)	Constructs a datagram socket and binds it to any available port on the local host machine.
receive(DatagramPacket p)	Receives a datagram packet from this socket.
send(DatagramPacket p)	Sends a datagram packet from this socket.
DatagramPacket(byte[] buf, int length)	Constructs a DatagramPacket for receiving packets of length.
DatagramPacket(byte[] buf,int length, InetAddress address, int port)	Constructs a datagram packet for sending packets of length to the specified port number on the specified host.
getData()	Returns the data buffer

Program:**Client:**

```
import java.net.*;
import java.io.*;
import java.lang.*;
public class ex15client
{
    public static void main(String args[]) throws IOException
    {
        byte[] buff=new byte[512];
        DatagramSocket soc=new DatagramSocket();
        String s="HELLO SERVER-FROM CLIENT";
        buff=s.getBytes();
        InetAddress a=InetAddress.getLocalHost();
        DatagramPacket pac=new DatagramPacket(buff,buff.length,a,8888);
        soc.send(pac);
        System.out.println("End of Sending");
        byte[] buff1=new byte[512];
        pac=new DatagramPacket(buff,buff1.length);
        soc.receive(pac);
        String messg=new String(pac.getData());
        System.out.println(messg);
        System.out.println("End Of Sending ");
    }
}
```

Server:

```
import java.io.*;
import java.net.*;
import java.lang.*;
public class ex15server
{
    public static void main(String args[]) throws IOException
    {
        byte[ ] buff=new byte[512];
        DatagramSocket soc=new DatagramSocket(8888);
        DatagramPacket pac=new DatagramPacket(buff,buff.length);
        System.out.println("Server started ");

        soc.receive(pac);
        String msg=new String(pac.getData());
        System.out.println(msg);
        System.out.println("End of Reception");
        String s="FROM SERVER HELLO CLIENT";
        byte[ ] buff1=new byte[512];
        buff1 =s.getBytes();
```



```
InetAddress a=pac.getAddress();
int port=pac.getPort();
pac=new DatagramPacket(buff,buff1.length,a,port);
soc.send(pac);
System.out.println("End of Sending");
}
}
```

Sample Input & Output:

Server:

```
D:\Program Files\Java\jdk1.6.0_20\bin>javac ex15server.java
D:\Program Files\Java\jdk1.6.0_20\bin> java ex15server
```

```
Server started
HELLO SERVER-FROM CLIENT
End of Reception
End of Sending
```

Client:

```
D:\Program Files\Java\jdk1.6.0_20\bin>javac ex15client.java
D:\Program Files\Java\jdk1.6.0_20\bin> java ex15client
```

```
End of Sending
```

Result:

Thus the above Java Program was successfully executed and verified.

Ex.No: 16

Date:

Sliding Window Protocols

Aim:

To write a Java program to implement Sliding Window Protocols.

Class:

DataInputStream:

Class lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

Following are the important points about DataInputStream:

- i) An application uses a data output stream to write data that can later be read by a data input stream.
- ii) DataInputStream is not necessarily safe for multithreaded access. Thread safety is optional and is the responsibility of users of methods in this class.

PrintStream:

Class adds functionality to another output stream, the ability to print representations of various data values conveniently.

Program:

Sender:

```
import java.io.*;
import java.net.*;
import java.rmi.*;
public class sender
{
    public static void main(String a[])throws Exception
    {
        ServerSocket ser=new ServerSocket(10);
        Socket s=ser.accept();
        DataInputStream in=new DataInputStream(System.in);
        DataInputStream in1=new DataInputStream(s.getInputStream());
        String sbuff[]=new String[8];
        PrintStream p;
        int sptr=0,sws=8,nf,ano,i;
        String ch;
        do
        {
            p=new PrintStream(s.getOutputStream());
```

```

System.out.print("Enter The Number of Frames:");
nf=Integer.parseInt(in.readLine());
p.println(nf);
if(nf<=sws-1)
{
System.out.println("Enter "+nf+"Message To Be Send");
for(i=0;i<=nf;i++)
{
sbuff[sptr]=in.readLine();
p.println(sbuff[sptr]);
sptr=++sptr%8;
}
sws-=nf;
System.out.println("Acknowledgement Received");
ano=Integer.parseInt(in1.readLine());
System.out.println("for"+ano+"frames");
sws+=nf;
}
else
{
System.out.println("The Number Of Frames Exceeds Window Size");
break;
}
System.out.println("\n Do You Want To Send Some More Frames:");
ch=in.readLine();
p.println(ch);
}
while(ch.equals("yes"));
s.close();
}
}

```

Receiver:

```

import java.io.*;
import java.net.*;
public class receiver
{
public static void main(String a[])throws Exception
{
Socket s=new Socket(InetAddress.getLocalHost(),10);
DataInputStream in=new DataInputStream(s.getInputStream());
PrintStream p=new PrintStream(s.getOutputStream());
int i=0,rptra=-1,nf,rws=8;
String rbuff[]=new String[8];
String ch;
System.out.println();

```

```

do
{
nf=Integer.parseInt(in.readLine());
if(nf<=rws-1)
{
for(i=1;i<=nf;i++)
{
rptr=++rptr%8;
rbuff[rptr]=in.readLine();
System.out.println("The Received Frame "+rptr+"is:"+rbuff[rptr]);
}
rws-=nf;
System.out.println("\n Acknowledgement Send\n");
p.println(rptr+1);
rws+=nf;
}
else
break;
ch=in.readLine();
}
while(ch.equals("yes"));
}
}

```

Sample Input & Output:**Sender:**

```
C:\Program files\Java\jdk1.5.0_06\bin>javac sender.java
C:\Program files\Java\jdk1.5.0_06\bin>java sender
Enter The Number of Frames: 4
Enter 4 Message To Be Send
Hai..
Hello..
How Are You..?
Bye..!

Acknowledgement Received
For 4 frames

Do You Want To Send Some More Frames: No
```

Receiver:

```
C:\Program files\Java\jdk1.5.0_06\bin>javac receiver.java
C:\Program files\Java\jdk1.5.0_06\bin>java receiver

The Recieved Frame 0 is: Hai..
The Recieved Frame 1 is: Hello..
The Recieved Frame 2 is: How Are You..?
The Recieved Frame 3 is: Bye..!
The Recieved Frame 4 is:

Acknowledgement Send
```

Result:

Thus the above Java Program was successfully executed and verified.