

1(a) Lex program to recognize a valid arithmetic expression and to recognize the identifiers and operators present. Print them separately.

```
% {
    #include<stdio.h>
    int oprc=0, idc=0, digc=0, top=-1;
    char s[20];
% }
iden [a-zA-Z][a-zA-Z0-9]*
opr  [+/*]
digit [0-9]+
%%
[ \n\t]+      {;}
['(']         { s[++top]='('; }
[')']         {
    if(s[top]=='(' && top!=-1)
        top--;
    else
    {
        printf("\nInvalid Expression.\n");
        exit(0);
    }
}
{iden}        { idc++;}
{digit}       { digc++;}
{opr}         { oprc++;}
.             {
                                printf("\nInvalid Expression.\n");
                                exit(0);
                                }
%%
main()
{
    system("clear");
    printf("Enter an Arithmetic Expression:\n");
    yylex();
    if(((idc+digc)==oprc+1)&&top== -1)
    {
        printf("\nValid Expression.\n");
        printf("\nNumber of Operators= %d\nNumber of Identifiers= %d\nNumber of
            Digits= %d\n", oprc, idc, digc);
    }

    else
```

```
        printf("\nInvalid Expression.\n");
    }
    yywrap()
    {}
```

Execution:

```
[root@cse1bldea ssw]# lex 2a.lex
[root@cse1bldea ssw]# cc lex.yy.c
[root@cse1bldea ssw]# ./a.out
```

Output:

I.

Enter an Arithmetic Expression:

$((a+b)-c*9)/2$

Valid Expression.

Number of Operators= 4

Number of Identifiers= 3

Number of Digits= 2

II.

Enter an Arithmetic Expression:

$((a+b)-c$

Invalid Expression.

1(b) YACC program to evaluate an arithmetic expression involving operators +, -, * and /.

```
% {
    #include<stdio.h>
    #include<stdlib.h>
% }
%token num
%%
START: expr '\n' {printf("\nValue= %d\n",$$); exit(0);}
      | error {printf("\nInvalid Expression.\n"); exit(0);}
      ;
expr: expr '+' term {$$=$1+$3;}
     | expr '-' term {$$=$1-$3;}
     | term
     ;
term: term '*' factor {$$=$1*$3;}
     | term '/' factor {$$=$1/$3;}
     | factor
     ;
factor: '(' expr ')' {$$=$2;}
       | num
       ;
%%
main()
{
    system("clear");
    printf("Enter an Arithmetic Expression to Evaluate:\n");
    yyparse();
}
yyerror()
{}
yylex()
{
    int c;
    c=getchar();
    if(isdigit(c))
    {
        yylval=c-'0';
        return num;
    }
    return c;
}
```

Execution:

```
[root@cse1bldea ssw]# yacc -d 5a.yacc  
[root@cse1bldea ssw]# cc y.tab.c  
[root@cse1bldea ssw]# ./a.out
```

Output:

I.

Enter an Arithmetic Expression to Evaluate:

$((2+3)*5)+5$

Value= 30

II.

Enter an Arithmetic Expression to Evaluate:

$(2++3)$

Invalid Expression.

2. YACC program to recognize the grammar ($a^n b$) (where $n \geq 10$).

```
% {
    #include<stdio.h>
    #include<stdlib.h>
% }
%token A B NEWLINE
%%
START: VALID {printf("\nString is Accepted.\n"); exit(0);}
      | error {printf("\nString is Rejected.\n"); exit(0);}
      ;
VALID: S NEWLINE
      ;
      S: X B
      ;
      X: A A A A A A A A A A
      |A X
      ;
%%
main()
{
    system("clear");
    printf("Enter the string of a and b:\n");
    yyparse();
}
yyerror()
{ }
```

Supporting Lex Program:

```
% {
    #include<stdio.h>
    #include"y.tab.h"
% }
%%
[a]    {return A;}
[b]    {return B;}
[\n]   {return NEWLINE;}
%%
yywrap()
{ }
```

Execution:

```
[root@cse1bldea ssw]# lex 6.lex  
[root@cse1bldea ssw]# yacc -d 6.yacc  
[root@cse1bldea ssw]# cc y.tab.c lex.yy.c  
[root@cse1bldea ssw]# ./a.out
```

Output:

I.

Enter the string of a and b:

aab

String is Rejected.

II.

Enter the string of a and b:

aaaaaaaaaab

String is Accepted.

3. Design, develop and implement YACC/C program to construct *Predictive / LL(1) Parsing Table* for the grammar rules: $A \rightarrow aBa$, $B \rightarrow bB \mid \epsilon$. Use this table to parse the sentence: *abba*\$.

```
#include<stdio.h>
#include<conio.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 productions in a grammar\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]=='\n')
{
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]);
fo1[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;
}
}
```

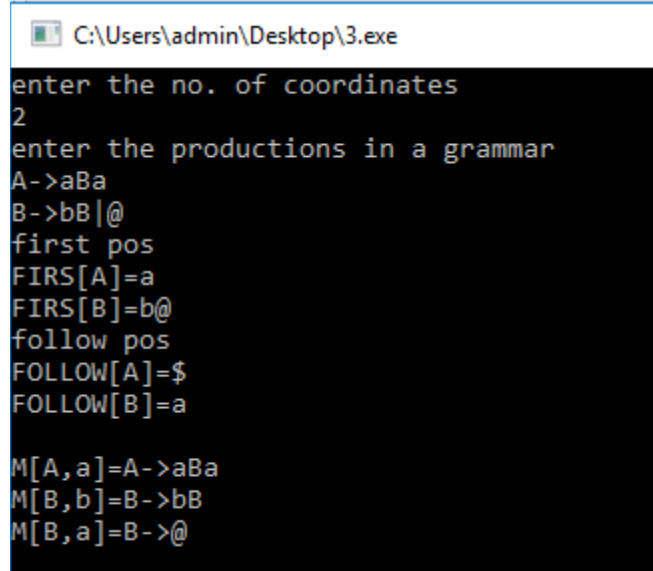
```
}
}
else
{
a1:c=0;
a=0;
while(st[k][0]!=st[a][0])
{
a++;
}
while((fol[a][c]!='\0')&&(st[a][0]!=st[i][0]))
{
for(m=0;m<l;m++)
{
if(fol[i][m]==fol[a][c])
goto q1;
}
fol[i][l]=fol[a][c];
l++;
q1:c++;
}
}
goto k1;
}
fol[i][l]='\0';
}
printf("follow pos\n");
for(i=0;i<n;i++)
printf("FOLLOW[%c]=%s\n",st[i][0],fol[i]);
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]=='\n')
j++;
}
}
getch();
}
```

Output:



```
C:\Users\admin\Desktop\3.exe
enter the no. of coordinates
2
enter the productions in a grammar
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->@
```

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

```
#include<stdio.h>
#include<conio.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 \n E->E*E \n E->(E) \n E->id");
puts("enter 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$s\t%s\t%s\t%sid",stk,a,act);
check();
}
else
{
stk[i]=a[j];
stk[i+1]='\0';
a[j]=' ';
printf("\n$s\t%s\t%s\t%ssymbols",stk,a,act);
check();
}
}
getch();
}
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$s%s\t%s$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$s%s\t%s$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$s%s\t%s$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$s%s\t%s$s\t%s",stk,a,ac);
i=i-2;
}
}
```

Output:

```
C:\Users\admin\Desktop\Untitled2.exe
GRAMMAR is E->E+E
E->E*E
E->(E)
E->id
enter 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 E_
```

*****YACC program to recognize a valid arithmetic expression that uses operators +, -, * and /.**

```
% {
    #include<stdio.h>
% }
%token ID DIG LB RB SUB MUL DIV ADD
%left SUB ADD
%left DIV MUL
%%
START: VALID {printf("\nValid Expression.\n"); exit(0);}
      | error {printf("\nInvalid Expression.\n"); exit(0);}
      ;
VALID: E
      ;
      E: E ADD T
        | E SUB T
        | T
        ;
      T: T MUL F
        | T DIV F
        | F
        ;
      F: ID
        | DIG
        | LB E RB
        ;
%%
main()
{
    system("clear");
    printf("Enter an Arithmetic Expression:\n");
    yyparse();
}
yyerror()
{}
```


Supporting Lex Program:

```
% {  
    #include "y.tab.h"  
% }  
%%  
[0-9]+          {return DIG;}  
[a-zA-Z]+[a-zA-Z0-9]* {return ID;}  
[(]             {return LB;}  
[)]             {return RB;}  
[\-]            {return SUB;}  
[\*]            {return MUL;}  
[/]             {return DIV;}  
[+]             {return ADD;}  
%%  
yywrap()  
{ }
```

Execution:

```
[root@cse1bldea ssw]# lex 4a.lex  
[root@cse1bldea ssw]# yacc -d 4a.yacc  
[root@cse1bldea ssw]# cc y.tab.c lex.yy.c  
[root@cse1bldea ssw]# ./a.out
```

Output:

I.

Enter an Arithmetic Expression:

((a+b)-c)*d

Valid Expression.

II.

Enter an Arithmetic Expression:

((a+b)

Invalid Expression.

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],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,"\nDIV 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);
getch();
}
```

Output:

input.txt

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

output.txt

```
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
```

6(a) Lex program to count the numbers of comment lines in a given C program. Also eliminate them and copy the resulting program into separate file.

```
% {
    #include<stdio.h>
    int i=0;
% }
id [a-zA-Z0-9 \n\t]*
%%
"/*" {id} "*" {i++;}
%%
main()
{
    system("clear");
    FILE *fp;
    fp=fopen("abc.c","r");
    yyin=fp;
    yylex();
    printf("Number of Comment Lines= %d\n",i);
}
yywrap()
{}
```

Supporting abc.c file:

```
/*This is a C Program*/
main()
{
    int x=4; /*Declaration of x*/
    int y=7,z; /*Declaration of y and z*/
    z=x+y;
    printf("ans= %d",z);
}
```

Execution:

```
[root@cse1bldea ssw]# lex 1b.lex
[root@cse1bldea ssw]# cc lex.yy.c
[root@cse1bldea ssw]# ./a.out
```

Output:

```
main()
{
    int x=4;
    int y=7,z;
    z=x+y;
    printf("ans= %d",z);
}
```

Number of Comment Lines= 3

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

Lex File

```
% {
#include <stdio.h>
#include "y.tab.h"
extern yylval;
% }
%%
[ \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 File

```
% {
#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>
extern int yylex();
extern int yyparse();
extern FILE *yyin;
main() {
FILE *myfile = fopen("sam_input.c", "r");
if (!myfile) {
printf("I can't open sam_input.c!");
return -1;
}
```

```
yyin = myfile;
do {
    yyparse();
} while (!feof(yyin));
printf("numbers = %d\nKeywords = %d\nIdentifiers = %d\noperators = %d\n",
    dig, key, id, op);
}
void yyerror() {
    printf("EEK, parse error! Message: ");
    exit(-1);
}
```

Output :

Input file

```
1 void main()
2 {
3     float a123;
4     char a;
5     char b123;
6     char c;
7     if (sum == 10)
8         printf("pass");
9     else
10        printf("fail");
11 }
```

```
admin1@admin1-HP-ProDesk-400-G3-DM:~$ ./a.out
keyword is void
identifier is main

keyword is float
identifier is a123

keyword is char
identifier is a

keyword is char
identifier is b123

keyword is char
identifier is c

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 = 10
operators = 2
admin1@admin1-HP-ProDesk-400-G3-DM:~$
```


- 7. Design, develop and execute a program in C/C++ to simulate the working of Shortest Remaining time and Round Robin Scheduling algorithms. Experiment with different Quantum sizes for the Round Robin algorithm. In all cases, determine the average turn Around time. Input can be read from keyboard or from a file.**

```
#include<stdio.h>
#include<conio.h>
void roundrobin();
void srtf();
main()
{
    int choice;
    printf("Enter the choice \n");
    printf(" 1. Round Robin\n 2. SRTF\n 3. Exit \n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            printf("Round Robin scheduling algorithm\n");
            roundrobin();
            break;
        case 2:
            printf("\n \n ---SHORTEST REMAINING TIME NEXT---\n \n ");
            srtf();
            break;
        case 3: exit(0);
    }
} //end of main

void roundrobin()
{
    int n,bt[10],st[10],tat[10],tq,stat=0;
    int i,j,k,count,sq=0,temp;
    float atat=0.0;
    printf("Enter number of processes:\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter burst time for process :",i);
        scanf("%d",&bt[i]);
        st[i]=bt[i]; //service time
    }
    printf("\n enter time quantum:");
    scanf("%d",&tq);
```

```
while(1)
{
    for(i=0,count=0;i<n;i++)
    {
        temp=tq;
        if(st[i]==0) // when service time of a process equals zero then count value is incremented
        {
            count++;
            continue;
        }
        if(st[i]>tq) // when service time of a process greater than time quantum then time
            st[i]=st[i]-tq; //quantum value subtracted from service time
        else
            if(st[i]>=0)
            {
                temp=st[i]; // temp1 stores the service time of a process
                st[i]=0; // making service time equals 0
            }
            sq=sq+temp; // utilizing temp1 value to calculate turnaround time
            tat[i]=sq; // turn around time
        } //end of for

        if(n==count) // it indicates all processes have completed their task because the count value
            break; // incremented when service time equals 0
    } //end of while

    for(i=0;i<n;i++) // to calculate the turnaround time of each process
    {
        stat=stat+tat[i]; // summation of turnaround time
    }

    atat=(float)stat/n; // average turnaround time
    printf("Process_no Burst time Turn around time\n");
    for(i=0;i<n;i++)
        printf("%d\t%d\t%d\n",i+1,bt[i],tat[i]);
    printf("Avg turn around time is %f\n",atat);
    getch();
} // end of Round Robin
```

```
void srtf()
{
    struct srtf1
    {
        int job;
        int bt;
```

```
};
struct srtf1 s[10],temp;
int tat[10],stat=0,i,j,n;
float atat=0.0;
printf("\n Enter the Number of processes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
    printf("\n Enter process no and Burst Time:");
    scanf("%d%d",&s[i].job,&s[i].bt);
}
//sorting based on size
for(i=0;i<n;i++)
    for(j=i+1;j<n;j++)
    {
        if(s[i].bt>s[j].bt)
        {
            temp=s[i];
            s[i]=s[j];
            s[j]=temp;
        }
    }
}

//turnaround time calculation
tat[0]=s[0].bt;
for(i=1;i<n;i++)
{
    tat[i]=tat[i-1]+s[i].bt;
}
//total(sum) turnaround time
for(i=0;i<n;i++)
{
    stat+=tat[i];
}
atat=(float)stat/n;
printf("\n \tJob\tBurst Time\t tat");
for(i=0;i<n;i++)
    printf("\n\t%d \t%d \t\t\t",s[i].job,s[i].bt,tat[i]);
printf("\n Total Turnaround Time : %d",stat);
printf("\n Average ttat : %f",atat);
getch();
}
```

Output:

```
$cc rrsrtf.c
$./a.out
```

1. Round Robin
2. SRTF
3. Exit

Enter the choice:1

Round Robin scheduling algorithm

Enter number of processes:2

Enter burst time for process 1 :2

Enter burst time for process 2 :3

enter time quantum:1

Process_no	Burst time	Turnaround time
------------	------------	-----------------

1	2	3
---	---	---

2	3	5
---	---	---

Avg turn around time is 4.000000

```
$./a.out
```

1. Round Robin
2. SRTF
3. Exit

Enter the choice:2

---SHORTEST REMAINING TIME NEXT---

Enter the Number of processes:2

Enter process no and Burst Time:1 3

Enter process no and Burst Time:2 2

Job	Burst Time	ttat
-----	------------	------

2	2	2
---	---	---

1	3	5
---	---	---

Total Turnaround Time : 7

Average ttat : 3.500000

8. Design, develop and run a program to implement the Banker's Algorithm. Demonstrate its working with different data values.

```
# include<stdio.h>
//# include<conio.h>
# include<stdlib.h>
# include<string.h>
# define MAX 10
struct process
{
char pid[4];
int maxneed[3];
int altd[3];
int need[3];
int finish;
}p[MAX];
int avl[3];
char safe[10][4];
void bankers(int n);
void main()
{
int n,i;
//clrscr();
printf("How many processes?");
scanf("%d",&n);
printf("Enter processes");
for(i=0;i<n;i++)
{
printf("\nEnter pid:");
scanf("%s",&p[i].pid);
printf("Enter maxneed:");
scanf("%d%d%d",&p[i].maxneed[0],&p[i].maxneed[1],&p[i].maxneed[2]);
printf("Enter allocated:");
scanf("%d%d%d",&p[i].altd[0],&p[i].altd[1],&p[i].altd[2]);
p[i].finish=0;
}
printf("Enter available resources:");
scanf("%d%d%d",&avl[0],&avl[1],&avl[2]);
bankers(n);
//getch();
}
void bankers(int n)
{
int i,k=0;
int j=0,m;
```

```
int count=0;
m=n*n;
for(i=0;i<n;i++)
{
p[i].need[0]=p[i].maxneed[0]-p[i].altd[0];
p[i].need[1]=p[i].maxneed[1]-p[i].altd[1];
p[i].need[2]=p[i].maxneed[2]-p[i].altd[2];
}
while(k<=m)
{
for(i=0;i<n;i++)
{
if(p[i].finish!=1 && p[i].need[0]<=avl[0] && p[i].need[1]<=avl[1] && p[i].need[2]<=avl[2])
{
strcpy(safe[j],p[i].pid);
j++;
count++;
p[i].finish=1;
avl[0]+=p[i].altd[0];
avl[1]+=p[i].altd[1];
avl[2]+=p[i].altd[2];
}
else
{
continue;
}
}
k++;
}
if(count==n)
{
printf("\nState is safe that is:\n");
for(i=0;i<n;i++)
{
printf("%s",safe[i]);
}
}
else
{
printf("\n\t !deadlock occur");
}
}
```

Output:

```
$ ./a.out
How many processes?5
Enter processes
Enter pid:p1
Enter maxneed:7 5 3
Enter allocated:0 1 0

Enter pid:p2
Enter maxneed:3 2 2
Enter allocated:2 0 0

Enter pid:p3
Enter maxneed:9 0 2
Enter allocated:3 0 2

Enter pid:p4
Enter maxneed:2 2 2
Enter allocated:2 1
1

Enter pid:p5
Enter maxneed:4 3 3
Enter allocated:0 0 2
Enter available resources:3 3 2

State is safe that is:
p2p4p5p1p3
```

```
$ ./a.out
How many processes?5
Enter processes
Enter pid:P1
Enter maxneed:7 5 3
Enter allocated:0 1 0

Enter pid:P2
Enter maxneed:3 2 2
Enter allocated:2 0 0

Enter pid:P3
Enter maxneed:9 0 2
Enter allocated:3 0 2

Enter pid:P4
Enter maxneed:2 2 2
Enter allocated:2 1 1

Enter pid:P5
Enter maxneed:4 3 3
Enter allocated:0 0 2
Enter available resources:3 3 0

!deadlock occurred
```


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()
{
    int ch,YN=1,i,l,f;
    char F[10],s[25];
    printf("\n\n\tEnter 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\n\t2:LRU \n\n\t4: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 4:
                exit(0);
        }
        printf("\n\n\tDo u want to continue IF YES PRESS 1\n\n\tIF NO PRESS 0 : ");
    }
```

```
scanf("%d",&YN);
}while(YN==1);return(0);
}
//FIFO
void FIFO(char s[],char F[],int l,int f)
{
int i,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;
}
}
//LRU
void lru(char s[],char F[],int l,int f)
{
int i,j=0,k,m,flag=0,cnt=0,top=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;
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:

```
Enter the no of empty frames: 3  
Enter the length of the string: 5  
Enter the string: hello  
***** MENU *****  
1:FIFO  
2:LRU  
4:EXIT  
  
Enter your choice: 1  
PAGE FRAMES FAULTS  
h h Page-fault 0  
e h e Page-fault 1  
l h e l Page-fault 2  
l h e l No page-fault  
o o e l Page-fault 3  
Do u want to continue IF YES PRESS 1  
IF NO PRESS 0 : 1  
***** MENU *****  
1:FIFO  
2:LRU  
4:EXIT  
Enter your choice: 2  
PAGE FRAMES FAULTS  
h h Page-fault 0  
e h e Page-fault 1  
l h e l Page-fault 2  
l h e l No page fault  
o e l o Page-fault 3  
Do u want to continue IF YES PRESS 1  
IF NO PRESS 0 : 1  
***** MENU *****  
1:FIFO  
2:LRU  
4:EXIT  
Enter your choice: 4
```

10. a) Design, develop and implement a C/C++/Java program to simulate a *numerical calculator*

```
#include <stdio.h>
void main()
{
    char operator;
    float num1, num2, result;
    printf("Simulation of a Simple Calculator\n");
    printf("*****\n");
    printf("Enter two numbers \n");
    scanf("%f %f", &num1, &num2);
    fflush(stdin);
    printf("Enter the operator [+,-,*,/] \n");
    scanf("%s", &operator);
    switch(operator)
    {
        case '+': result = num1 + num2;
        break;
        case '-': result = num1 - num2;
        break;
        case '*': result = num1 * num2;
        break;
        case '/': result = num1 / num2;
        break;
        default : printf("Error in operationn");
        break;
    }
    printf("\n %5.2f %c %5.2f = %5.2f\n", num1, operator, num2, result);
}
```

Output:

Simulation of a Simple Calculator

Enter two numbers

2

3

Enter the operator [+,-,*,/]

+

2.00 + 3.00 = 5.00

b) Design, develop and implement a C/C++/Java program to simulate *page replacement technique*

```
#include<stdio.h>
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;
void getData()
{
printf("\nEnter length of page reference sequence:");
scanf("%d",&n);
printf("\nEnter the page reference sequence:");
for(i=0; i<n; i++)
scanf("%d",&in[i]);
printf("\nEnter no of frames:");
scanf("%d",&nf);
}
void initialize()
{
pgfaultcnt=0;
for(i=0; i<nf; i++)
p[i]=9999;
}
int isHit(int data)
{
hit=0;
for(j=0; j<nf; j++)
{
if(p[j]==data)
{
hit=1;
break;
}
}
return hit;
}
int getHitIndex(int data)
{
int hitind;

for(k=0; k<nf; k++)
{
if(p[k]==data)
{
```

```
hitind=k;
break;
}
}
return hitind;
}
void dispPages()
{
for (k=0; k<nf; k++)
{
if(p[k]!=9999)
printf(" %d",p[k]);
}
}
void dispPgFaultCnt()
{
printf("\nTotal no of page faults:%d",pgfaultcnt);
}
void fifo()
{
initialize();
for(i=0; i<n; i++)
{
printf("\nFor %d :",in[i]);
if(isHit(in[i])==0)
{
for(k=0; k<nf-1; k++)
p[k]=p[k+1];
p[k]=in[i];
pgfaultcnt++;
dispPages();
}
else
printf("No page fault");
}
dispPgFaultCnt();
}

void optimal()
{
initialize();
int near[50];
for(i=0; i<n; i++)
{
printf("\nFor %d :",in[i]);
if(isHit(in[i])==0)
```

```
{
for(j=0; j<nf; j++)
{
int pg=p[j];
int found=0;
for(k=i; k<n; k++)
{
if(pg==in[k])
{
near[j]=k;
found=1;
break;
}
else
found=0;
}
if(!found)
near[j]=9999;
}
int max=-9999;
int repindex;
for(j=0; j<nf; j++)
{
if(near[j]>max)
{
max=near[j];
repindex=j;
}
}
p[repindex]=in[i];
pgfaultcnt++;
dispPages();
}
else

printf("No page fault");
}
dispPgFaultCnt();
}
void lru()
{
initialize();
int least[50];
for(i=0; i<n; i++)
{
printf("\nFor %d :",in[i]);
```



```
if(isHit(in[i])==0)
{
for(j=0; j<nf; j++)
{
int pg=p[j];
int found=0;
for(k=i-1; k>=0; k--)
{
if(pg==in[k])
{
least[j]=k;
found=1;
break;
}
else
found=0;
}
if(!found)
least[j]=-9999;
}
int min=9999;
int repindex;
for(j=0; j<nf; j++)
{
if(least[j]<min)
{
min=least[j];
repindex=j;
}
}
p[repindex]=in[i];
pgfaultcnt++;

dispPages();
}
else
printf("No page fault!");
}
dispPgFaultCnt();
}
void lfu()
{
int usedcnt[100];
int least,repin,sofarcnt=0,bn;
initialize();
for(i=0; i<nf; i++)
```

```
usedcnt[i]=0;
for(i=0; i<n; i++)
{
printf("\n For %d :",in[i]);
if(isHit(in[i]))
{
int hitind=getHitIndex(in[i]);
usedcnt[hitind]++;
printf("No page fault!");
}
else
{
pgfaultcnt++;
if(bn<nf)
{
p[bn]=in[i];
usedcnt[bn]=usedcnt[bn]+1;
bn++;
}
else
{
least=9999;
for(k=0; k<nf; k++)
if(usedcnt[k]<least)
{
least=usedcnt[k];
repin=k;
}

p[repin]=in[i];
sofarcnt=0;
for(k=0; k<=i; k++)
if(in[i]==in[k])
sofarcnt=sofarcnt+1;
usedcnt[repin]=sofarcnt;
}
dispPages();
}
}
dispPgFaultCnt();
}
void secondchance()
{
int usedbit[50];
int victimptr=0;
initialize();
```

```
for(i=0; i<nf; i++)
usedbit[i]=0;
for(i=0; i<n; i++)
{
printf("\nFor %d:",in[i]);
if(isHit(in[i]))
{
printf("No page fault!");
int hitindex=getHitIndex(in[i]);
if(usedbit[hitindex]==0)
usedbit[hitindex]=1;
}
else
{
pgfaultcnt++;
if(usedbit[victimptr]==1)
{
do
{
usedbit[victimptr]=0;
victimptr++;
if(victimptr==nf)
victimptr=0;
}
while(usedbit[victimptr]!=0);
}
if(usedbit[victimptr]==0)
{
p[victimptr]=in[i];
usedbit[victimptr]=1;
victimptr++;
}
dispPages();
}
if(victimptr==nf)
victimptr=0;
}
dispPgFaultCnt();
}
int main()
{
int choice;
while(1)
{
```

```
printf("\nPage Replacement Algorithms\n1.Enter data\n2.FIFO\n3.Optimal\n4.LRU\n5.LFU\n6.Second Chance\n7.Exit\nEnter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
getData();
break;
case 2:
fifo();
break;
case 3:
optimal();
break;
case 4:
lru();
break;
case 5:
lfu();
break;
case 6:
secondchance();
break;
default:
return 0;
break;
}
}
```

Output:

Page Replacement Algorithms

1.Enter data

2.FIFO

3.Optimal

4.LRU

5.LFU

6.Second Chance

7.Exit

Enter your choice:1

Enter length of page reference sequence:8

Enter the page reference sequence:2

3

4

2

3

5

6

2

Enter no of frames:3

Page Replacement Algorithms

1.Enter data

2.FIFO

3.Optimal

4.LRU

5.LFU

6.Second Chance

7.Exit

Enter your choice:2

For 2 : 2

For 3 : 2 3

For 4 : 2 3 4

2 :No page fault

For 3 :No page fault

For 5 : 3 4 5

For 6 : 4 5 6

For 2 : 5 6 2

Total no of page faults:6

Page Replacement Algorithms

1.Enter data

2.FIFO

3.Optimal

4.LRU

5.LFU

6.Second Chance

7.Exit

Enter your choice:3

For 2 : 2

For 3 : 2 3

For 4 : 2 3 4

For 2 :No page fault

For 3 :No page fault

For 5 : 2 5 4

For 6 : 2 6 4

For 2 :No page fault

Total no of page faults:5

Page Replacement Algorithms

1.Enter data

2.FIFO

3.Optimal

4.LRU

5.LFU

6.Second Chance

7.Exit

Enter your choice:4

For 2 : 2

For 3 : 2 3

For 4 : 2 3 4

For 2 :No page fault!

For 3 :No page fault!

For 5 : 2 3 5

For 6 : 6 3 5

For 2 : 6 2 5

Total no of page faults:6

Page Replacement Algorithms

1.Enter data

2.FIFO

3.Optimal

4.LRU

5.LFU

6.Second Chance

7.Exit

Enter your choice:5

For 2 : 2

For 3 : 2 3

For 4 : 2 3 4

For 2 :No page fault!

For 3 :No page fault!

For 5 : 2 3 5

For 6 : 2 3 6

For 2 :No page fault!

Total no of page faults:5

Page Replacement Algorithms

1.Enter data

2.FIFO

3.Optimal

4.LRU

5.LFU

6.Second Chance

7.Exit

Enter your choice:7