Exercises

1. a)Find sum of N numbers. Allocate the memory dynamically to the numbers.
   b)Find the following for a matrix. Use the concept of pointer to 2D array.
   i) sum of principal diagonal elements
   ii)sum of secondary diagonal elements
   iii)sum of all elements

2. Using array of pointer concept
   a) Find product of two Matrices
   b) Sort n names in alphabetical order

3. Implement and demonstrate the following C functions using pass-by-reference method.
   i)StrCopy()
   ii)StrConcat()
   iii)strcomp()
   iv)Strrev()

4. Define an EMPLOYEE structure with members Emp_name, Emp-id, Dept-name and Salary. Read and display data of N employees. Employees may belong to different departments. Write a function to find total salary of employees of a specified department. Use the concept of pointer to structure and allocate the memory dynamically to EMPLOYEE instances.

5. a) Define a recursive factorial function. Evaluate the following series for N terms using a function which takes x in degrees and a pointer to factorial function as parameters.
   $Sin(x)=x-x^3/3!+ x^5/5!-$ .....
   b) To copy the contents of one file to another, taking file names as command line arguments. Display the contents of target file on the screen.

6. Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and binary operators + - * /. Apply the concept of stack data structure to solve this problem.

7. Write a C program to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary operators. The operators are + - * and /.

8. Write recursive functions for the following and demonstrate their use.
   a) Binary Search
   b) Tower of Hanoi problem.

9. A Call center phone system has to hold the phone calls from customers and provide service based on the arrival time of the calls. Write a C program to simulate this system using appropriate data structure. Program should have options to add and remove the phone calls in appropriate order for their service.

10. Write a C program to simulate the working of a circular Queue of integers. Represent circular queue element as a structure and use array of structures as your implementation method. Start and end of the circular queue must be identified by an empty array element.

11. Write a program to create a singly linked list that maintains a list of names in alphabetical order. Implement the following operations on the list.
    a) Insert a new name
    b) Delete a specified name

12. Write a C program to maintain a stack of integers using linked implementation method

13. Write a C program to support the following operations on a doubly linked list
    a) Insert a new node to the left of the node whose key value is read as an input.
    b) Delete a node with given data, if it is found. otherwise display appropriate error message.

14. Write a C program
    a) To construct a binary search tree (BST) of integers.
    b) To traverse the tree using inorder, preorder and postorder traversal methods

15. A list of unordered numbers is given in a file. The file may have duplicate numbers. Read the numbers from the file, construct a binary tree of these numbers and display the numbers in ascending order.

```
/*1 a) Find sum of N numbers. Allocate the memory dynamically to the
numbers.*/

#include<stdio.h>
#include<malloc.h>
main()
{

int i,n;
float sum=0,*p;
        printf("enter n\n");
        scanf("%d",&n);

        p=(float*)malloc(n*sizeof(float));
        if(p==NULL)
        {
                printf("allocation failed\n");
                return;
        }
        printf("enter %d nos\n",n);
        for(i=0;i<n;i++)
                scanf("%f",(p+i));

        for(i=0;i<n;i++)
                sum=sum+ *(p+i);

        printf("sum=%f", sum);
}

OUTPUT:
enter n
5
enter 5 nos
2
3
4
5
6
sum=20.000000

/*1 b) Find the following for a matrix. Use the concept of pointer to 2D array.
```

i) Sum of principal diagonal elements
ii)sum of secondary diagonal elements
iii)sum of all elements*/

```
/*sum of primary & secondary diagonal of a matrix using array of pointers*/
#include<stdio.h>
#include<malloc.h>
void main()
{
int i,j,m,n,d1=0,d2=0, sum=0;
int (*p)[3];

printf("Enter row and column\n");
scanf("%d%d",&m,&n);

if(m!=n)
{
        printf("not a square matrix\n");
        return;
}

p=(int*)malloc(m*n*sizeof(int));

printf("Enter matrix elements\n");
for(i=0;i<m;i++)
         for(j=0;j<n;j++)
                 scanf("%d",*(p+i)+j);

for(i=0;i<m;i++)
         d1=d1+*(*(p+i)+i);

for(i=0,j=(n-1);i<m;i++,j--)
        d2=d2+*(*(p+i)+j);

for(i=0;i<m;i++)
        for(j=0;j<n;j++)
                sum+= *(*(p+i)+j));

printf("the primary matrix is = %d\n",d1);
printf("the secondary matrix is = %d\n",d2);
printf("the sum of all elements is = %d\n",sum);
```

}

OUTPUT:

1) Enter row and column
   2
   3
   not a square matrix

2) Enter row and column
   3
   3
   Enter matrix elements
   2
   3
   1
   4
   5
   6
   7
   8
   9
   the primary matrix is = 16
   the secondary matrix is = 13
   the sum of all elements is = 45

3) Enter row and column
   2
   2
   Enter matrix elements
   1
   1
   1
   1
   the primary matrix is = 2
   the secondary matrix is = 2
   the sum of all elements is = 4

/*2. Using array of pointer concept

a) Find product of two Matrices

b) Sort n names in alphabetical order*/

a)Find product of two Matrices

```
#include<stdio.h>
#include<malloc.h>
void main()
{
int i,j,k,m,n,p,q,sum;
int *A[3],*B[3],*C[3];
printf("enter the order of matrix 1\n");
scanf("%d%d",&m,&n);
printf("enter the order of matrix 2\n");
scanf("%d%d",&p,&q);
if(n!=p)
{
        printf("matrices cannot be multiplied\n");
        return;
}
for(i=0;i<m;i++)
{
          A[i]=(int*)malloc(n*sizeof(int));
         C[i]=(int*)malloc(q*sizeof(int));
}

for(i=0;i<p;i++)
{
        B[i]=(int*)malloc(q*sizeof(int));
}

printf("enter elements of matrix 1\n");
for(i=0;i<m;i++)
          for(j=0;j<n;j++)
                   scanf("%d",A[i]+j);

printf("enter elements of matrix 2\n");
for(i=0;i<n;i++)
          for(j=0;j<q;j++)
                  scanf("%d",B[i]+j);
for(i=0;i<m;i++)
        for(k=0;k<q;k++)
```

```
        {
                *(C[i]+k)=0;
                for(j=0;j<n;j++)
                        *(C[i]+k)= *(C[i]+k)+ *(A[i]+j)* *(B[j]+k);
          }
/*display*/

for(i=0;i<m;i++)
{
         for(j=0;j<q;j++)
                printf("%d\t",*(C[i]+j));

                printf("\n");
}
}
/*output*/
enter the order of matrix 1
2 3
enter the order of matrix 2
2 3
matrices cannot be multiplied
Press any key to continue
enter the order of matrix 1
2 3
enter the order of matrix 2
2 3
matrices cannot be multiplied
Press any key to continue
```

b)/*sorting of n names using array of pointers*/

```
#include<stdio.h>
#include<malloc.h>
#include<string.h>
```

```
void main()
{
 int i,j,n;
 char *names[10],temp[80];

printf("enter n\n");
 scanf("%d",&n);

 for(i=0;i<n;i++)
            names[i]=(char*)malloc(80*sizeof(char));

/*reading n names*/
 printf("enter %d names\n",n);
            getchar();

 for(i=0;i<n;i++)
             gets(names[i]);

/*bubble sort*/
 for(i=1;i<n;i++)
              for(j=0;j<(n-i);j++)
                     if(strcmp(names[j],names[j+1])>0)
                      {          temp=names[j];
                                 names[j]=names[j+1];
                                 names[j+1]=temp;
                      }

    /*display*/
 printf("sorted array of names \n");
 for(i=0;i<n;i++)
            puts(names[i]);
}
```

OUTPUT:

```
enter n
4
enter 4 names
ds
cd
bcc
md
```

sorted array of names
bcc
cd
ds
md

/*3. Implement and demonstrate the following C functions using pass-by-reference method.
i)StrCopy()
ii)StrConcat()
iii)strcomp()
iv)Strrev()*/

```
/*strcopy(),strcomp(), strconcat() functions using pointer parameters*/
#include<stdio.h>

/*strcopy()*/
void strcopy(char *s1,char *s2)
{
        while(*s2!='\0')
        {
                *s1=*s2;
                s1++;
                s2++;
        }
        *s1='\0';
 }

/*strcomp()*/
int strcomp(char *s1,char *s2)
{
        while(*s1!='\0'||*s2!='\0')
        {
                if(*s1!=*s2)
                        return(*s1-*s2);
                s1++;
                s2++;
        }
 return 0; /*strings are equal*/
}

/*strconcat()*/
void strconcat(char *s1,char *s2)
{
        while(*s1!='\0')
                s1++;
        while(*s2!='\0')
        {
                *s1=*s2;
                s1++;
                s2++;
        }
 *s1='\0';
}
```

```
void Strreverse(char *s1)
{
          int i=0,j,k=0;char s3[50];
           while(*s1)
           {
                    s1++,k++;
           }
           for(i=0,j=k-1;    j>=0;    j--,i++)
           {
                    --s1;
                      s3[i]=*s1;
           }
           s3[i]='\0';
           strcopy(s1,s3);
}

int main()
{
      char s1[80],s2[80],t1[80];
      int choice, result;

      printf("enter 2 strings\n");
      gets(s1);
      gets(s2);

      strcopy(t1,s1);
      printf("---------MENU----------\n");
      printf("\n1.strcomp\n2.strcopy\n3.strconcat\n4.Sting reverse\n");
      for(;;)
      {
           printf("enter your choice\n");
           scanf("%d",&choice);

           switch(choice)
           {
         case 1: strcopy(s1,t1);
                    result=strcomp(s1,s2);
                    if(result>0)
                              printf("%s is greater than %s\n",s1,s2);
```

```
                else if(result<0)
                        printf("%s is greater than %s\n",s2,s1);
                else
                       printf("strings are equal\n");
                break;


        case 2:  strcopy(s1,t1);
                 printf("strings before copying, s1=%s\t, s2=%s\n",s1,s2);
                strcopy(s1,s2);
                 printf("after copying, s1= %s\t, s2= %s\n",s1,s2);
                 break;


        case 3:  strcopy(s1,t1);
                printf("strings before concatenation, s1=%s\t , s2=%s\n",s1,s2);
                strconcat(s1,s2);
                 printf("after concatenation, s1= %s\t , s2= %s\n",s1,s2);
                 break;


        case 4:  strcopy(s1,t1);
                printf("strings before reversing, s1=%s\t , s2=%s\n",s1,s2);
                 printf(" strings after reversing\n");
                Strreverse(s1);
                 printf("s1=%s\n",s1);
                 strcopy(s1,s2);
                Strreverse(s1);
               printf("s2=%s\n",s1);
                break;

       default:  return;
        }
 }
}
```

OUTPUT:

```
enter 2 strings
nitte
meenakshi
---------MENU-----------
```

1.strcomp
2.strcopy
3.strconcat
4.Sting reverse
enter your choice
4
strings before reversing, s1=nitte , s2=meenakshi
strings after reversing
s1=ettin
s2=ihskaneem
enter your choice
3
strings before concatenation, s1=nitte , s2=meenakshi
after concatenation, s1= nittemeenakshi , s2= meenakshi
enter your choice
2
strings before copying, s1=nitte , s2=meenakshi
after copying, s1= meenakshi , s2= meenakshi
enter your choice
1
nitte is greater than meenakshi
enter your choice

/*4.Define an EMPLOYEE structure with members Emp_name, Emp-id, Dept-name and Salary. Read and display data of N employees. Employees may belong to different departments. Write a function to find total salary of employees of a specified department. Use the concept of pointer to structure and allocate the memory dynamically to EMPLOYEE instances. */

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct employee{
```

```
                                        char emp_name[20];
                                        int emp_id;
                                        char dept_name[20];
                                        float salary;
                                        };
compute_total_sal(struct employee *e,char dn[], int m);
void main()
{
        int m,i;
        struct employee *emp;
        char dname[20],choice;
        printf("Enter the number of employees:");
        scanf("%d",&m);
        emp=(struct employee *)malloc(m*sizeof(struct employee));
        /*Read employee details*/
        for(i=0;i<m;i++)
        {
                printf("Enter the details of Employee %d\n", i+1);
                printf("Enter the employee name:");
                scanf("%s",(emp+i)->emp_name);
                printf("Enter the employee ID:");
                scanf("%d",&(emp+i)->emp_id);
                printf("Enter the employee department:");
                scanf("%s",&(emp+i)->dept_name);
                printf("Enter the salary of employee:");
                scanf("%f",&(emp+i)->salary);
        }
        /* Print employee details*/
        printf("************************Employee Details********************** \n");

        printf("%-15s%-25s%-15s%-15s\n","ID","EMPLOYEE NAME","DEPARTMENT","SALARY");
        for(i=0;i<m;i++)
        {
        printf("%-15d%-25s%-15s%-15g \n",(emp+i)->emp_id,(emp+i)->emp_name,(emp+i)->dept_name,
(emp+i)->salary);

         }
        do{
                printf("\n Enter the department for which the total salary has to be computed:");
                scanf("%s",dname);
                compute_total_sal(emp,dname,m);
                printf("\n Do you want to continue[Y/N]:");
                scanf(" %c",&choice);
        }while(choice=='Y' || choice=='y');
}

/*find total salary of employees of a specified department. */
```

```
compute_total_sal(struct employee *e,char dn[], int m)
{
	float totalsal=0;
	int i,flag=0;
	for(i=0; i<m;i++)
		if(strcmp((e+i)->dept_name,dn)==0)
		{flag=1;
		totalsal += (e+i)->salary;}
		if(flag==0)
			printf("\n No such department\n");
		else
			printf("\n Total salary of employees in department %s is %f",dn,totalsal);
}

/*
Sample Output
---------------
Enter the number of employees:3
Enter the details of Employee 1
Enter the employee name:Anil
Enter the employee ID:3333
Enter the employee department:CSE
Enter the salary of employee:30000
Enter the details of Employee 2
Enter the employee name:Banu
Enter the employee ID:7777
Enter the employee department:HR
Enter the salary of employee:15000
Enter the details of Employee 3
Enter the employee name:John
Enter the employee ID:5555
Enter the employee department:CSE
Enter the salary of employee:20000

**********Employee Details**********

*/output*/
Enter the number of employees:3
Enter the details of Employee 1
Enter the employee name:vinutha
Enter the employee ID:111
Enter the employee department:computers
Enter the salary of employee:40000
Enter the details of Employee 2
Enter the employee name:divya
Enter the employee ID:222
```

```
Enter the employee department:electronics
Enter the salary of employee:39000
Enter the details of Employee 3
Enter the employee name:neeraj
Enter the employee ID:333
Enter the employee department:computers
Enter the salary of employee:60000
**********Employee Details**********
ID          EMPLOYEE NAME          DEPARTMENT     SALARY
111          vinutha                    computers           40000
222          divya                      electronics          39000
333          neeraj                     computers           60000

 Enter the department for which the total salary has to be computed:computers

 Total salary of employees in department computers is 100000.000000
 Do you want to continue[Y/N]:y

 Enter the department for which the total salary has to be computed:electronics

 Total salary of employees in department electronics is 39000.000000
 Do you want to continue[Y/N]:y

 Enter the department for which the total salary has to be computed:mechanical

 No such department

 Do you want to continue[Y/N]:
```

```
/*5.
a) Define a recursive factorial function.
Evaluate the following series for N terms using a function
which takes x in degrees and a pointer to factorial function as parameters.
  Sin(x)=x-x^3/3!+ x^5/5!- .....
  */

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
```

```
double eval_sinx(double xrad,long int NT, long int (*funcptr)());
long int fact(long int n);

#define pi 22/7

void main()
{
	long int NT;
	double sinx, xrad, xdeg;

	printf("Enter the angle in degrees:");
	scanf("%lf",&xdeg);

	printf("Enter the number of terms:");
	scanf("%ld",&NT);

	xrad=xdeg*pi/180;
	sinx=eval_sinx(xrad,NT,fact);
	printf("\n Sin(%lf)=%lf\n",xdeg, sinx);
}

double eval_sinx(double xrad,long int NT, long int (*funcptr)())
{
	int i;
	double sinx=0;
	int sign=1;
	for(i=1;  i<=NT;  i+=2)
	{
		sinx=sinx + sign * pow(xrad,i)/(*funcptr)(i);
		sign=-(sign);
	}
	return(sinx);
}

long int fact(long int n)
{
	if(n==0 || n==1)
		return(1);
	return(n*fact(n-1));
}

/*Sample output
------------------
Enter the angle in degrees:90
Enter the number of terms:10
```

```
 Sin(90.000000)=1.000003

Enter the angle in degrees:45
Enter the number of terms:10

Sin(45.000000)=0.707330

Enter the angle in degrees:60
Enter the number of terms:15

 Sin(60.000000)=0.866236
Press any key to continue

*/
```

/*5B) To copy the contents of one file to another, taking file names as command line arguments. Display the contents of target file on the screen.*/

```
#include<stdio.h>
void main(int argc,char *argv[])
        {
                char c;
                FILE *fpt1,*fpt2;
                if(argc<3)
                        {
```

```
                printf("File name not provided As command line arguments\n");
                return;
        }
    else
        {
                fpt1=fopen(argv[1],"r");
                fpt2=fopen(argv[2],"w");
                if(fpt1==NULL)
                {
                        printf("\n File not found");
                        return;
                }
                else{
                        do
                                {
                                        c=fgetc(fpt1);
                                        fputc(c,fpt2);
                                }while(c!=EOF);

                        }
                printf("\n Source file copied to target file");
                fclose(fpt1);
                fclose(fpt2);
        }
    fpt2=fopen(argv[2],"r");
    printf("\n The contents of the file after copying are:\n");
    while((c=fgetc(fpt2))!=EOF)
        printf("%c",c);
}

/*Sample output
------------------
C:/>gedit sample.text
Hello ! Welcome to NMIT, Department of CSE.!!!

c:/>./a.out sample.txt newfile.txt
Source file copied to target file
The contents of the file after copying are:
Hello ! Welcome to NMIT, Department of CSE.!!!

C:/> gedit newfile.txt
Hello ! Welcome to NMIT, Department of CSE.!!!
*/
```

Fo practicing:

Define a structure STUDENT with members Name and USN. Write a C program to construct a stack data structure of N STUDENT objects and to perform the following operations on it:

a) PUSH-To add a new student to the stack
b) POP---- To remove a student from the stack

*/

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define SIZE 5

struct student{
                                char name[20];
                                char USN[10];
            };
struct stack{
            struct student stud[SIZE];
                        int top;
             };


void push(struct stack *ps,  struct student st1);
struct student pop(struct stack *ps);

void main()
{
        struct stack s;
        struct student st1,rt1;
        int choice,i;
        s.top=-1;

        do{
          printf("\n 1:PUSH\t 2:POP\t 3:DISPLAY\t 4:QUIT");
          printf("\n Enter your choice:");
          scanf("%d",&choice);

switch(choice)
          {
          case 1: printf("Enter the name and USN of student to push:");
                        scanf("%s %s",st1.name,st1.USN);
                        push(&s,st1);
                        break;

          case 2: rt1=pop(&s);
```

```
                        printf("The student popped is %s with USN %s\n",rt1.name,rt1.USN);
                        break;

        case 3: if(s.top==-1)
                                printf("\n Stack empty");
                else
                {
                                printf("Stack contents are:\n");
                                for(i=s.top;i>=0;i--)
                                {
                                        printf("%s %s\n",s.stud[i].name,s.stud[i].USN);
                                }
                  }
                                break;

        case 4:         printf("QUITTING OPERATION STACK .....\n");
                 break;

        default :printf("No such option\n");
        }

        }while(choice!=4);
}


void push(struct stack *ps,  struct student st1)
{
        if(ps->top= =SIZE-1)
                printf("Stack Overflow\n");
        else
        {
                ++(ps->top);
                strcpy(ps->stud[ps->top].name,  st1.name);
                strcpy(ps->stud[ps->top].USN,   st1.USN);
        }
}


struct student pop(struct stack *ps)
{
        struct student r;
        if(ps->top = = -1)
        {
                printf("\n Stack Underflow");
                exit(1);
        }
```

```
                strcpy(r.name,   ps->stud[ps->top].name);
                strcpy(r.USN,   ps->stud[ps->top].USN);
                (ps->top)--;
                return(r);
}


/* Sample Output
------------------

 1:PUSH  2:POP   3:DISPLAY      4:QUIT
 Enter your choice:1
 Enter the name and USN of student to push: Roshni 1NT15CS121

 1:PUSH  2:POP   3:DISPLAY      4:QUIT
 Enter your choice:1
 Enter the name and USN of student to push:Avinash 1Nt15CS001

 1:PUSH  2:POP   3:DISPLAY      4:QUIT
 Enter your choice:1
 Enter the name and USN of student to push:Shubham 1NT14CS072

 1:PUSH  2:POP   3:DISPLAY      4:QUIT
 Enter your choice:3
 Stack contents are:
Shubham 1NT14CS072?
Avinash 1Nt15CS001Shubham
Roshni 1NT15CS121Avinash

 1:PUSH  2:POP   3:DISPLAY      4:QUIT
 Enter your choice:2
The student popped is Shubham with USN 1NT14CS072¦¦Shubham

 1:PUSH  2:POP   3:DISPLAY      4:QUIT
 Enter your choice:2
Press any key to continue */
/*6. Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix
expression. The expression consists of single character operands and binary operators + - * /. Apply the concept
of stack data structure to solve this problem.*/


 /*Infix to postfix conversion*/

#include<stdio.h>
char stack[50];
int top=  -1;
```

```
/*push()*/
void push(char ch)
{           stack[++top]=ch;
}

/*pop()*/
char pop()
{           return(stack[top- -]);
}

/*prcd()*/
int prcd(char ch)
{          int p;
           switch(ch)
            {           case '$':
                        case '^':         p=3;
                                          break;

                        case '*':
                        case '/':         p=2;
                                          break;

                      case '+':
                        case '-':         p=1;
                                          break;

                        case '(':         p=-1;
                                          break;
                     }
  return p;
}

/*conversion()*/
void conversion(char infix[],char postfix[])
{                   int i=0,p=0;
                    char ch;

                    while((ch=infix[i])!='\0')
                    {
```

```
                switch(ch)
                {
                        default :       postfix[p++]=ch;
                                        break;

                        case '(':       push(ch);
                                        break;

                        case ')':       while(top!= -1&& stack[top]!='(')
                                                postfix[p++]=pop();
                                        pop();  /*discard ( */
                                        break;

                        case '*':
                        case '/':
                        case '+':
                        case '-':        while(top!= -1  && prcd(stack[top]) >= prcd(ch))
                                                postfix[p++]=pop();
                                         push(ch);
                                         break;
                        case '$':
                        case '^':       /*associativity right to left*/
                                        while(top!= -1 && prcd(stack[top]) > prcd(ch))
                                                postfix[p++]=pop();
                                        push(ch);
                                        break;
                    }
   i++;
  }

 while (top!= -1)
          postfix[p++] = pop();
  postfix[p]='\0';
}

int main()
{                       char infix[50],postfix[50];
                        printf("enter valid infix expression\n");
                        scanf("%s", infix);
                        conversion(infix, postfix);
                        printf("postfix expression= %s\n", postfix);
```

}

OUTPUT:
enter valid infix expression
(a+b)*(c-d)/e$f
postfix expression= ab+cd-*ef$/

/*7.Write a C program to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary operators. The operators are + - * and /.*/

/*Evaluation of postfix expression*/
#include<stdio.h>
#include<math.h>
#include<ctype.h>
float stack[50];
int top=-1;

```
void push(float n)
{
        stack[++top]=n;
}
/*pop()*/
float pop()
{
        return(stack[top--]);
}


/*evaluate()*/
float eval(char postfix[])
{
            float op1,op2,res;
            char ch;
            int i=0;
            while((ch=postfix[i])!='\0')
            {
                  if(isdigit(ch))
                        push(ch-'0');
                  else
                  {
                        op2=pop();
                        op1=pop();
                        switch(ch)
                        {
                              case '$':
                              case '^': res=pow(op1,op2);
                                    break;
                               case '*': res=op1*op2;
                                    break;
                               case '/': res=op1/op2;
                                    break;
                               case '+': res=op1+op2;
                                    break;
                               case '-': res=op1-op2;
                                    break;
                          }
                         push(res);
                   }
```

```
                i++;
        }
        return(pop());
}
int main()
{
        char postfix[50];
        float res;

        printf("Enter postfix expression\n");
        scanf("%s",postfix);

        res=eval(postfix);

        printf("Result=%g\n",res);
}
```

OUTPUT:

1) Enter postfix expression
   23+43-*
   Result=5

2) Enter postfix expression
   231$$
   Result=8

/*8. Write recursive functions for the following and demonstrate their use.
a) Binary Search
b) Tower of Hanoi problem.*/

/*RECURSIVE BINARY SEARCH*/

```
#include<stdio.h>
#include<stdlib.h>

/*binary search*/
int R_bin_search(int a[], int key, int low, int high)
{
        int mid;
```

```
        if(low>high) return -1;

         mid=(low+high)/2;

        if(key==a[mid])
                return mid;

        if(key<a[mid])
                return(R_bin_search(a,key,low,(mid-1)));

        return(R_bin_search(a,key,(mid+1),high));
}

void main()
{
  int key,*a,i,n,res, repeat, p;

  do{
                printf("ENTER n\n");
                scanf("%d",&n);

                a=(int*)malloc(n*sizeof(int));

                printf("ENTER THE NUMBERS\n");
                 for(i=0;i<n;i++)
                        scanf("%d",(a+i));

        //checking whether it is sorted.
        p=0;
        for(i=0;i<n-1;i++)
                if(a[i]<a[i+1])p++;

        if(p= =n-1)
                printf("Yes. It is sorted in ascending order\n");
       else
         {
                printf("Input is not sorted.Enter numbers in ascending order\n");
                goto AGAIN;
        }
```

```
        printf("ENTER THE KEY TO BE SEARCHED\n");
        scanf("%d",&key);

        res=R_bin_search(a,key,0,(n-1));
        if(res = = -1)
                printf("KEY NOT FOUND\n");
        else
                 printf("%d FOUND AT LOCATION %d\n",key,(res+1));

AGAIN:
                printf("Press 1 to continue\n");
                scanf("%d",&repeat);
    }while(repeat= =1);
}

ENTER n
3
ENTER THE NUMBERS
12
23
45
Yes. It is sorted in ascending order
ENTER THE KEY TO BE SEARCHED
12
12 FOUND AT LOCATION 1
Press 1 to continue
1
ENTER n
2
ENTER THE NUMBERS
23
5
Input is not sorted. Enter numbers in ascending order
Press 1 to continue
5
```

/*C PROGRAMS TO IMPLEMENT TOWERS OF HANOI*/

```
#include<stdio.h>
/*towers*/
void towers(int n, char src, char dest, char aux)
{
        if(n= =1)
        {
                printf("MOVE DISK 1 FROM PEG %c TO PEG %c\n",src,dest);
                return;
        }
        towers(n-1,src,aux,dest);
        printf("MOVE DISK %d FROM PEG %c TO PEG %c\n",n,src,dest);
        towers(n-1,aux,dest,src);
}
```

```
void main()
{
        int n;
        printf("ENTER THE NUMBER OF DISKS\n");
        scanf("%d",&n);

        printf("MOVES MADE\n");
        towers(n,'A','C','B');
}
```

OUTPUT:

```
1) ENTER THE NUMBER OF DISKS
  1
  MOVES MADE
  MOVE DISK 1 FROM PEG A TO PEG C
2) ENTER THE NUMBER OF DISKS
  2
  MOVES MADE
  MOVE DISK 1 FROM PEG A TO PEG B
  MOVE DISK 2 FROM PEG A TO PEG C
  MOVE DISK 1 FROM PEG B TO PEG C
3) ENTER THE NUMBER OF DISKS
  3
  MOVES MADE
  MOVE DISK 1 FROM PEG A TO PEG C
  MOVE DISK 2 FROM PEG A TO PEG B
  MOVE DISK 1 FROM PEG C TO PEG B
  MOVE DISK 3 FROM PEG A TO PEG C
  MOVE DISK 1 FROM PEG B TO PEG A
  MOVE DISK 2 FROM PEG B TO PEG C
  MOVE DISK 1 FROM PEG A TO PEG C
```

/*9A Call center phone system has to hold the phone calls from customers and provide service based on the arrival time of the calls. Write a C program to simulate this system using appropriate data structure. A Program should have options to add and remove the phone calls in appropriate order for their service. */

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 10

int rpt=1,q[SIZE];
int ch;
int frnt,rear;

void main()
{
	frnt=rear=-1;
	while(rpt)
	{
		printf("A Call center phone system using static QUEUE\n");
		printf("select a operation from the followings:\n");
```

```
		printf("1:ADD incoming call\n2:REMOVE the call for service\n3:Display pending
calls\n4:EXIT\n");
		scanf("%d",&ch);

		switch(ch)
		{
			case 1: insert();
			break;
			case 2: del();
			break;
			case 3: show();
			break;
			case 4: printf("END\n"); exit(0);
			break;
			default: printf("Please enter correct choice\n");
		}
		printf(" \nTo continue press non zero digit:\n");
		scanf("%d",&rpt);
	}
}

int insert()
{
	int n;

	if(rear==(SIZE-1))
	{
		printf("ALREADY QUEUE IS FULL:\n");
		return;
	}

	printf("Enter the call ID to be inserted to the system:\n");
	scanf("%d",&n);

	if(frnt==-1&&rear==-1)
	{
		rear++;frnt++;
		*(q+rear)=n;
		printf("success\n");
	}
	else
	{
		rear++;
		*(q+rear)=n;
		printf("success\n");
	}
```

```
        return;
}

int  del()
{
        int temp;
        if(frnt==-1)
        {
                printf("no calls waiting in the queue:\n");
                return;
        }
        temp=*(q+frnt);
        frnt++;
        printf("Call answered is %d\n",temp);

        if(frnt > rear)
        frnt=rear=-1;

        return;
}

int show()
{
        int i;
        if(frnt==-1 && rear==-1)
        {
                printf("Sorry no pending calls\n");
                return;
        }
        printf("Here is the waiting queue of calls:\n");
        printf("FRONT\t");
        for(i=frnt;i<=rear;i++)
                printf("%d\t",*(q+i));
        printf("REAR\n");
        return;
}
```

OUTPUT:

```
A Call center phone system using static QUEUE
select a operation from the followings:
1:ADD incoming call
2:REMOVE the call for service
3:Display pending calls
4:EXIT
1
Enter the call ID to be inserted to the system:
```

```
2
success

To continue press non zero digit:
1
A Call center phone system using static QUEUE
select a operation from the followings:
1:ADD incoming call
2:REMOVE the call for service
3:Display pending calls
4:EXIT
1
Enter the call ID to be inserted to the system:
5
success

To continue press non zero digit:
1
A Call center phone system using static QUEUE
select a operation from the followings:
1:ADD incoming call
2:REMOVE the call for service
3:Display pending calls
4:EXIT
1
Enter the call ID to be inserted to the system:
8
success

To continue press non zero digit:
1
A Call center phone system using static QUEUE
select a operation from the followings:
1:ADD incoming call
2:REMOVE the call for service
3:Display pending calls
4:EXIT
3
Here is the waiting queue of calls:
FRONT        2        5        8        REAR

To continue press non zero digit:
1
A Call center phone system using static QUEUE
select a operation from the followings:
1:ADD incoming call
2:REMOVE the call for service
```

```
3:Display pending calls
4:EXIT
2
Call answered is 2

To continue press non zero digit:
1
A Call center phone system using static QUEUE
select a operation from the followings:
1:ADD incoming call
2:REMOVE the call for service
3:Display pending calls
4:EXIT
3
Here is the waiting queue of calls:
FRONT        5        8        REAR
 */
```

/* 10. Write a C program to simulate the working of a circular Queue of integers. Represent circular queue element as a structure and use array of structures as your implementation method. Start and end of the circular queue must be identified by an empty array element.*/

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 5

struct item{
        int ele;
};

struct cqueue{
         struct item it[MAXSIZE];
         int front;
         int rear;
        };

void insertitem(struct cqueue *pq,int x);
int deleteitem(struct cqueue *pq);

void main()
```

```
{
struct cqueue cq;
int x,choice,r;
cq.front=MAXSIZE-1,cq.rear=MAXSIZE-1;
do{
        printf(".............MENU.............");
        printf("\n 1:INSERT\t 2:REMOVE\t 3:QUIT\n");
        printf("Enter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1: printf("Enter the item to be inserted:");
                        scanf("%d",&x);
                        insertitem(&cq,x);
                        break;
        case 2: r=deleteitem(&cq);
                        printf("The item deleted is %d\n",r);
                        break;
        case 3: printf("QUITTING OPERATION QUEUE .........\n");
                    break;
        default :printf("No such option\n");
        }
        }while(choice!=3);
}
void insertitem(struct cqueue *pq,int x)
{
        if(pq->rear==MAXSIZE-1)
                pq->rear=0;
        else
                (pq->rear)++;
        if(pq->rear==pq->front)
        {
                printf("Circular Queue Overflow\n");;
                exit(1);
        }
        pq->it[pq->rear].ele=x;
        return;
}
int deleteitem(struct cqueue *pq)
{
        if(pq->front==pq->rear)
        {
                printf("Queue underflow\n");
                exit(1);
        }
        if(pq->front==MAXSIZE-1)
```

```
                pq->front=0;
        else
                (pq->front)++;
        return (pq->it[pq->front].ele);
}

/*Sample output
-----------------
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:1
Enter the item to be inserted:10
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:1
Enter the item to be inserted:40
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:1
Enter the item to be inserted:20
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:1
Enter the item to be inserted:70
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:2
The item deleted is 10
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:2
The item deleted is 40
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:1
Enter the item to be inserted:8
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:2
The item deleted is 20
.............MENU.............
 1:INSERT       2:REMOVE       3:QUIT
Enter your choice:3
QUITTING OPERATION QUEUE .........

*/
```

/*11Write a program to create a singly linked list that maintains a list of names in alphabetical order. Implement the following operations on the list.
a) Insert a new name
b) Delete a specified name*/

```
#include<stdio.h>
#include<stdlib.h>

struct stud
{
        int id;
        char name[10];
        int sem;
        struct stud *next;
};

typedef struct stud NODE;

NODE *head;
NODE* insert_f();

void main()
```

```
{
	int rpt=1;
	int ch;

	while(rpt)
	{
		printf("select a singly linked list operation from the followings:\n");
		//printf("1:insert at the front\n2:delete at the end\n3:Display\n4:EXIT\n");
		printf("1: Insert a new name \n2: Delete a specified name \n3:Display\n4:EXIT\n");

		scanf("%d",&ch);

	switch(ch)
	{
		case 1: insert_f();
		break;
		case 2: del_e();
		break;
		case 3: show();
		break;
		case 4: printf("END\n"); exit(0);
		break;
		default: printf("please enter correct choice\n");
	}
	printf(" \nTo continue press non zero digit:\n");
	scanf("%d",&rpt);
	}
}

//end of main()-------------------------------------------------------------

NODE* insert_f()
{
	NODE *n, *m, *temp1, *temp2;
	n=(NODE *)malloc(sizeof(NODE));

	printf("enter student's data in order:\n");
	printf("student ID\n");
	scanf("%d",&n->id);

	printf("student name\n");
	scanf("%s",n->name);

	printf("semester:\n");
	scanf("%d",&n->sem);
```

```
        n->next=NULL;

if(head==NULL)

        head=n;

else if(strcmp(head->name, n->name)>0)
{
        n->next=head;
        head=n;
}
else
{
        for(temp1=NULL, temp2=head;
                temp2!= NULL && strcmp(n->name,temp2->name)>0;
                        temp2=temp2->next)
                        {
                        temp1=temp2;
                        }

        temp1->next=n;
        n->next=temp2;
}
return(head);
}

//---end of insert()-----------------------------------------------
del_e()
{
        NODE *temp1,*temp2;
        char name[10];
        int flag=0;
        if(head==NULL)
        {
                printf("empty\n");
                return;
        }

        printf("enter student name for deletion\n");
        scanf("%s",name);

        if(strcmp(head->name,name)==0)
        {
                head=head->next;
                printf("deleted\n");
        }
```

```
        else
        {
                for(temp1=NULL,temp2=head;temp2!=NULL;temp2=temp2->next)
                {
                        if(strcmp(name,temp2->name)==0)
                        {
                                flag=1;
                                break;
                        }
                        else
                                temp1=temp2;
                }

        if(flag==0)
        {
                printf("Student name %s 's record not present in the list\n",name);
                return;
        }

        temp1->next=temp2->next;
        printf("%s student's data has been deleted\n",temp2->name);
        free(temp2);
}
return;
}
//-end of delete()-----------------------------------------------------

show()
{
        NODE *N;
        if(head==NULL)
        {
                printf("EMPTY\n");
                return;
        }

        printf("%-15s%-20s%-10s\n",”NAME”,”ID “,”SEM");
        N=head;

        while(N!=NULL)
        {
                printf(("%-15s%-20d%-10d\n”,N->name,N->id,N->sem);
                N=N->next;
        }
```

```
        return;
}
//---------------------------------------------------------------------
```
OUTPUT
```
select a singly linked list operation from the followings:
1:insert at the front
2:delete at the end
3:Display
4:EXIT
1
enter student's data in order:
student ID
23
student name
sham
semester:
2

To continue press non zero digit:
1
select a singly linked list operation from the followings:
1:insert at the front
2:delete at the end
3:Display
4:EXIT
1
enter student's data in order:
student ID
2
student name
ram
semester:
4

To continue press non zero digit:
1
select a singly linked list operation from the followings:
1:insert at the front
2:delete at the end
3:Display
4:EXIT
1
enter student's data in order:
student ID
6
student name
amar
```

```
semester:
2

To continue press non zero digit:
1
select a singly linked list operation from the followings:
1:insert at the front
2:delete at the end
3:Display
4:EXIT
3
NAME          ID      SEM
Amar          6       2
ram           2       4
sham          23      2

To continue press non zero digit:
1
select a singly linked list operation from the followings:
1:insert at the front
2:delete at the end
3:Display
4:EXIT
2
enter student name for deletion
ram
ram student's data has been deleted

To continue press non zero digit:
1
select a singly linked list operation from the followings:
1:insert at the front
2:delete at the end
3:Display
4:EXIT
3
NAME      ID    SEM
amar      6     2
sham      23    2

To continue press non zero digit:
*/
```

12. Write a C program to maintain a stack of integers using linked implementation method

```
/*implementation of dynamic stack using singly linked list*/
#include<stdio.h>
#include<malloc.h>
#define size 3
int count=0;
/*node structure*/
struct node
{
  int info;
  struct node *link;
};
typedef struct node* NODE;
NODE first=NULL;
/*getnode()*/
NODE getnode()
{
  NODE temp;
  temp=(NODE)malloc(sizeof(struct node));
  if(temp==NULL)
```

```
  {
    printf("allocation failed\n");
    return;
  }
  return temp;
}
/*push function()*/
void push(int item)
{
  NODE temp;
  temp=getnode();
  temp->info=item;
  temp->link=NULL;
  if(count==size)
  {
    printf("STACK OVERFLOW\n");
    return;
  }
  /*empty stack*/
  if(first==NULL)
    first=temp;
  else
  {
    temp->link=first;
    first=temp;
  }
  count++;
}
/*pop function()*/
void pop()
{
  NODE temp=first;
  if(count==0)
  {
    printf("STACK UNDERFLOW\n");
    return;
  }

    printf("item deleted = %d\n",first->info);
          first=first->link;
          free(temp);
```

```
  count--;
}
/*display()*/
void display()
{
  NODE cur=first;
  if(first==NULL)
  {
    printf("STACK EMPTY\n");
    return;
  }
  printf("stack contents\n");
  while(cur!=NULL)
  {
    printf("%d\t",cur->info);
    cur=cur->link;
  }
}
void main()
{
  int choice,data;
  printf("_ _ _ _ _MENU_ _ _ _ _\n");
  printf("1.PUSH\t2.POP\t3.DISPLAY\n");
  for(;;)
  {
    printf("\nenter the choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
      case 1: printf("enter the element to be pushed\n");
              scanf("%d",&data);
              push(data);
              break;
      case 2: pop();
              break;
      case 3: display();
              break;
      default: return;
    }
  }}
```

OUTPUT:

```
_ _ _ _ _MENU_ _ _ _ _
1.PUSH          2.POP  3.DISPLAY

enter the choice
1
enter the element to be pushed
5

enter the choice
1
enter the element to be pushed
6

enter the choice
1
enter the element to be pushed
7

enter the choice
1
enter the element to be pushed
8
STACK OVERFLOW

enter the choice
3
stack contents
7         6         5
enter the choice
2
item deleted = 7

enter the choice
2
item deleted = 6

enter the choice
2
item deleted = 5
```

```
enter the choice
2
STACK UNDERFLOW

enter the choice
3
STACK EMPTY
enter the choice
```

/*13Create Doubly linked list by inserting at the front of list. Node entry is an integer. Operations are: inset new node to the left of the node who's key value is read as an input and delete the node of a given data. If not found display a message.*/

```
#include<stdio.h>
#include<stdlib.h>

struct dll
{
        int num;
        struct dll *left,*right;
};

typedef struct dll NODE;

NODE *head=NULL;

void main()
{
int rpt=1;
int ch,data;

        printf("Enter integer to create doubly linked list\n");
        scanf("%d",&data);

do{
        create(data);
        scanf("%d",&data);
```

```
}while(data!=999);
show();
        while(rpt)
        {
        printf("\nSelect a doubly linked list operation from the followings:\n");
        printf("1:Inset new node to the left of the node whose key value is read as an input\n");
        printf("2:Delete the node of a given data\n3:Display\n4:EXIT\n");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1: insert();
                break;
                case 2: del();
                break;
                case 3: show();
                break;
                case 4: printf("END\n"); exit(0);
                break;
                default: printf("please enter correct choice\n");
        }

        printf(" \nTo continue press non zero digit:\n");
        scanf("%d",&rpt);
        }
}
//end of main()--------------------------------------

create(int d)
{
        NODE *temp;
        temp=(NODE *)malloc(sizeof(NODE));

        temp->num=d;
        temp->left = temp->right=NULL;

        if(head==NULL)
        {
                head=temp;return;
        }

        temp->right=head;
        head->left=temp;
        head=temp;

        return;
}
//------------------------------------------------------
```

```
insert()
{
	NODE *new,*temp,*temp2;
	int key,flag=0;
	new=(NODE *)malloc(sizeof(NODE));

	printf("Enter an integer of new node\n");
	scanf("%d",&new->num);

	new->left = new->right=NULL;

	if(head==NULL)
	{
		head=new;return;
	}

	printf("enter the key value of existing node of the list\n");
	scanf("%d",&key);

	if(key==head->num)
	{
		head->left=new;
		new->right=head;
		head=new;
	}

	else
	{
		for(temp=head->right;temp!=NULL;temp=temp->right)
		{
			if(temp->num==key)
			{
				flag=1;
				break;
			}
		}

		if(flag==1)
		{
			temp2=temp->left;

			new->left=temp2;
			new->right=temp;

			temp2->right=new;
			temp->left=new;
```

```
                }
                else
                        printf("give proper input\n");

        }
return;
}

//---end of insert()---------------------------------------
del()
{
        NODE *temp;
        int flag=0,key;

        if(head==NULL)
        {
                printf("empty\n");
                return;
        }

        printf("enter the key value of existing node of the list\n");
        scanf("%d",&key);

        if(key==head->num)
        {
                temp=head;
                head=head->right;

                if(head!=NULL)
                        head->left=NULL;
                free(temp);
        }

        else
        {

                for(temp=head->right;temp!=NULL;temp=temp->right)

                        if(temp->num==key)
                        {
                                flag=1;
                                break;
                        }

                if(flag==1)
                {
```

```
                        if(temp->right==NULL) //deleting last node
                                temp->left->right=NULL;
                        else
                        {
                                temp->left->right=temp->right;
                                temp->right->left=temp->left;
                        }
                        printf("Node with key value %d has been deleted\n",key);
                        free(temp);
                }
                else
                        printf("Node not found in the list\n");

        }

return;
}

//-end of delete()---------------------------------------

show()
{
        NODE *temp;

        if(head==NULL)
                printf("EMPTY\n");
        else
        {
                printf("Dougly linked list is\nSTART<---->");

                for(temp=head;temp!=NULL;temp=temp->right)
                        printf("%d<---->",temp->num);

                printf("END\n");
        }
return;
}
//-------------------------------------------------------

/*OUTPUT:
Enter integest to create doubly linked list
12
34
45
1
13
```

```
999
Dougly linked list is
START<---->13<---->1<---->45<---->34<---->12<---->END

Select a doubly linked list operation from the followings:
1:Inset new node to the left of the node whos key value is read as an input
2:Delete the node of a given data
3:Display
4:EXIT
3
Dougly linked list is
START<---->13<---->1<---->45<---->34<---->12<---->END

To continue press non zero digit:
1

Select a doubly linked list operation from the followings:
1:Inset new node to the left of the node whos key value is read as an input
2:Delete the node of a given data
3:Display
4:EXIT
3
Dougly linked list is
START<---->13<---->1<---->45<---->34<---->12<---->END

To continue press non zero digit:
1

Select a doubly linked list operation from the followings:
1:Inset new node to the left of the node whos key value is read as an input
2:Delete the node of a given data
3:Display
4:EXIT
2
enter the key value of existing node of the list
34
Node with key value 34 has been deleted

To continue press non zero digit:
1

Select a doubly linked list operation from the followings:
1:Inset new node to the left of the node whos key value is read as an input
2:Delete the node of a given data
3:Display
4:EXIT
3
```

Dougly linked list is
START<---->13<---->1<---->45<---->12<---->END

To continue press non zero digit:

*/

/*15.Write a C program
a) To construct a binary search tree (BST) of integers.
b) To traverse the tree using inorder, preorder and postorder traversal methods*/

```
/*BINARY SEARCH TREE*/
#include<stdio.h>
#include<malloc.h>
/*node structure*/
struct node
{
  int info;
  struct node *left;
  struct node *right;
};
typedef struct node* NODE;
/*getnode()*/
NODE getnode()
{
  NODE temp;
  temp=(NODE)malloc(sizeof(struct node));
  if(temp==NULL)
  {
    printf("allocation failed\n");
    return;
  }
  return temp;
}
/*creating a tree*/
```

```
NODE create(NODE root,int item)
{
  NODE temp,cur,suc;
  temp=getnode();
  temp->info=item;
  temp->left=NULL;
  temp->right=NULL;
  /*empty tree*/
  if(root==NULL)
  {
    root=temp;
    return root;
  }
  cur=root;
  suc=root;
  while(suc!=NULL)
  {
    cur=suc;
    if(item<cur->info)
      suc=suc->left;
    else
      suc=suc->right;
  }
  if(item<cur->info)
    cur->left=temp;
  else
    cur->right=temp;
  return root;
}
/*inodrer traversal*/
void inorder(NODE root)
{
  if(root!=NULL)
  {
    inorder(root->left);
    printf("%d\t",root->info);
    inorder(root->right);
  }
}
/*preorder traversal*/
void preorder(NODE root)
```

```
{
  if(root!=NULL)
  {
    printf("%d\t",root->info);
    preorder(root->left);
    preorder(root->right);
  }
}
/*postorder traversal*/
void postorder(NODE root)
{
  if(root!=NULL)
  {
    postorder(root->right);
    postorder(root->left);
    printf("%d\t",root->info);
  }
}
void main()
{
  NODE root=NULL;
  int choice,item;
  printf("\n__________MENU_________\n");
  printf("1.CREATE \t 2.INORDER\t3.PREORDER\t4.POSTORDER\n");
  for(;;)
  {
    printf("\nenter choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
      case 3: printf("preorder traversal\n");
              preorder(root);
              break;
      case 2: printf("inorder traversal\n");
              inorder(root);
              break;
      case 4: printf("postorder traversal\n");
              postorder(root);
              break;
      case 1: printf("enter item\n");
              scanf("%d",&item);
```

```
            root=create(root,item);
            break;
      default: return;
    }
  }
}
```

```
__________MENU_________
1.CREATE      2.INORDER  3.PREORDER  4.POSTORDER

enter choice
1
enter item
10

enter choice
1
enter item
6

enter choice
1
enter item
12

enter choice
1
enter item
9

enter choice
1
enter item
15

enter choice
2
inorder traversal
6        9        10       12       15
enter choice
3
```

```
preorder traversal
10      6       9       12      15
enter choice
4
postorder traversal
15      12      9       6       10
```

/*15. A list of unordered numbers is given in a file. The file may have duplicate numbers. Read the numbers from the file, construct a binary tree of these numbers and display the numbers in ascending order.*/

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int value;
   struct node *left, *right;
};


struct node *root;

struct node* insert(struct node* r, int data);

int main()
{
        root = NULL;
        int n, v, i;
        char in_name[80];
        FILE *in_file;
        printf("Enter file name:\n");
        scanf("%s", in_name);
        in_file = fopen(in_name, "r");
        if (in_file == NULL)
        {
                printf("Can't open %s for reading.\n", in_name);
                return 0;
        }
        else
        {
                while (fscanf(in_file, "%d",&v) ==1)
        root = insert(root, v);
        }
```

```
	fclose(in_file);

	printf("printing numbers in ascending order:\n");
	intrav(root);
	return 0;
}

struct node* insert(struct node* r, int data)
{
	if(r==NULL) // BST is not created created
	{
	r = (struct node*) malloc(sizeof(struct node)); // create a new node
	r->value = data;  // insert data to new node
	// make left and right childs empty
	r->left = NULL;
	r->right = NULL;
	}
	// if the data is less than node value then we must put this in left sub-tree
	else if(data==r->value)
		;
	else if(data < r->value)
	r->left = insert(r->left, data);

	// else this will be in the right subtree
	else
		r->right = insert(r->right, data);
	 return r;

}

intrav(struct node* r)
{
	if(r!=NULL)
	{
		intrav(r->left);		//Traverse left subtree.
		printf("%d\n",r->value);	 //Visit the root.
		intrav(r->right);		//Traverse right subtree.
	}
}

/*---------------------------------------------------------------------
OUTPUT:
Enter file name:
16_input.txt
12 34 5 78 12 34 90 32 10
printing numbers in ascending order:
```

5
10
12
32
34
78
90
*/