Recursion

1. Program to find GCD for given 2 nos.

```
#include<stdio.h>
int gcd(int,int);
void main()
{
    int  a,b,result;
    printf("enter any two numbers");
    scanf("%d %d",&a,&b);
    result=gcd(a,b);
    printf("gcd=%d",result);
}
int gcd(int a,int b)
{
    int r;
    r=a%b;
    if(r==0)
    {
        return b;
    }
    else
    {
        return gcd(b,r);
    }
}
```

Output:

```
enter any two numbers
15
18
gcd=3
```

2. Program to multiply 2 nos without using * operator

```
#include<stdio.h>
int mul(int x,int y)
{
    if(y==0)
    {
        return 0;
    }
    else if(y>0)
    {
        return (x+mul(x,y-1));
    }
    else if(y<0)
```

```
        {
            return -mul(x,-y);
        }
    }
    int main()
    {
        int a,b,result;
        printf("enter two numbers\n");
        scanf("%d %d",&a,&b);
        result=mul(a,b);
        printf("%d",result);
    }
```

Output:

```
enter two numbers
3
7
21
```

3.Program to divide 2 nos without using / operator

```
    #include<stdio.h>
    int division(int,int);
    void main()
    {

        int a,b,result;
        printf("enter two numbers\n");
        scanf("%d %d",&a,&b);
        result=division(a,b);
        printf("%d",result);

    }
    int division(int a,int b)
    {
        if(a-b<0)
            return 0;
        else if(a-b==0)
            return 1;
        else
            return division(a-b,b)+1;
    }
```

Output:

```
enter two numbers
250
15
16
```

4.Program to find power of a number (xy)

```
#include<stdio.h>
int power(int,int);
void main()
{
    int b,p,result;
    printf("enter base\n");
    scanf("%d",&b);
    printf("enter power\n");
    scanf("%d",&p);
    if(p>=0)
    {

        result=power(b,p);
        printf("%d^%d=%d",b,p,result);
    }

    if(p<0)
    {
        result=power(b,-p);
        printf("%d^%d=1/%d",b,p,result);
    }
}
int power(int b,int p)
{
    if(p!=0)
    {
        return (b*power(b,p-1));
    }
    else
    {
        return 1;

    }
}
```

Output:

```
enter base
2
enter power
6
2^6=64
```

5.Program to Check Whether a Number is Palindrome or Not

```
#include<stdio.h>
void main()
```

```
{
    int n,num ;
    printf("enter any number\n");
    scanf("%d",&n);
    num=reverse(n);
    if(n==num)
    {
        printf("palindrome");
    }
    else
    {
        printf("not palidrome");
    }
}
int reverse(int n)
{
    static sum=0;
    if(n!=0)
    {
        sum=sum*10+(n%10);
        reverse(n/10);
    }
    else
    {
        return sum;
    }
}
```

Output:

enter any number

121

Palindrome

6.Program for Tower of Hanoi problem

```
#include<stdio.h>
void toh(int n,char s,char d,char au)
{
    if(n==1)
    {
        printf("move disk %d from %c to %c\n",n,s,d);
        return;
    }
    toh(n-1,s,au,d);
    printf("move disk %d from %c to %c\n",n,s,d);
    toh(n-1,au,d,s);
```

```
}
void main()
{
    int n;
    printf("enter number of dics\n");
    scanf("%d",&n);
    toh(n,'A','C','B');
}
```

Output:

```
enter number of dics
3
move disk 1 from A to C
move disk 2 from A to B
move disk 1 from C to B
move disk 3 from A to C
move disk 1 from B to A
move disk 2 from B to C
move disk 1 from A to C
```

Stacks

1. Program to perform push, pop and peek operations on a stack using arrays

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define CAPACITY 5
int stack[CAPACITY],top=-1;
void push(int);
int isFull(void);
int pop(void);
int ifEmpty(void);
void peek(void);
void traverse(void);
void main()
{
    int op,ele,val;
    while(1)
    {
        printf("****MENU****\n");
```

```
        printf("1.push\n");
        printf("2.pop\n");
        printf("3.peek\n");
        printf("4.traverse\n");
        printf("5.quit\n");
        printf("enter your choice\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
                   printf("enter element\n");
                   scanf("%d",&ele);
                   push(ele);
                   break;
            case 2:
                   val=pop();
                   if(val==0)
                   {
                       printf("stack is empty\n");
                   }
                   else
                   {
                       printf("popped element %d\n",val);
                   }
                   break;
            case 3:
                   peek();
                   break;
            case 4:
                   traverse();
                   break;
            case 5:
                   exit(0);
            default:
                   printf("invalid choice\n");


        }
    }
}
void push(int ele)
{
    if(isFull())
    {
```

```
        printf("stack is overflow\n");
    }
    else
    {
        top++;
        stack[top]=ele;
        printf("%d is pushed\n",ele);
    }
}
int isFull()
{
    if(top==CAPACITY-1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int pop()
{
    if(isEmpty())
    {
        return 0;
    }
    else
    {
        return stack[top--];
    }
}
int isEmpty()
{
    if(top==-1)
    {
        return 1;

    }
    else
    {
        return 0;
    }
}
void peek()
```

```
{
    if(isEmpty())
    {
        printf("stack is empty\n");
    }
    else
    {
        printf("peek element is %d\n",stack[top]);
    }
}
void traverse()
{
    if(isEmpty())
    {
        printf("stack is empty\n");
    }
    else
    {
        int i;
        printf("elements in stack\n");
        for(i=0;i<=top;i++)
        {
            printf("%d\n",stack[i]);
        }
    }
}
```

Output:

```
****MENU****
1.push
2.pop
3.peek
4.traverse
5.quit
enter your choice
1
enter element
10
10 is pushed
****MENU****
1.push
2.pop
3.peek
4.traverse
5.quit
```

```
enter your choice
4
elements in stack
10
****MENU****
1.push
2.pop
3.peek
4.traverse
5.quit
enter your choice
5
```

2. programme to evaluate prefix expression

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<math.h>
#define CAPACITY 100
int top=-1;
int stack[CAPACITY];
int get_type(char);
void push(int);
int pop();
void main()
{
    char prefix[100];
    int len,val,op1,op2,result,i;
    printf("enter prefix expression\n");
    gets(prefix);
    len=strlen(prefix);
    for(i=len-1;i>=0;i--)
    {
        switch(get_type(prefix[i]))
        {
            case 0:
                    val=prefix[i]-'0';//characet     to     integer
conversion
                    push(val);
                    break;
            case 1:
                op1=pop();
                op2=pop();
                switch(prefix[i])
                {
```

```
                    case '+':
                           result=op1+op2;
                           break;
                    case '-':
                           result=op1-op2;
                           break;
                    case '*':
                           result=op1*op2;
                           break;
                    case '/':
                           result=op1/op2;
                           break;
                    case '%':
                           result=op1%op2;
                           break;
                    case '^':
                           result=pow(op1,op2);
                           break;
                }
                push(result);

        }
    }
    printf("RESULT:%d",stack[0]);


}
int get_type(char c)
{
    if(c=='+'||c=='-'||c=='*'||c=='/'||c=='%'||c=='^')
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
void push(int val)
{
    if(top==CAPACITY-1)
    {
        printf("stack overflow\n");
    }
```

```
    else
    {
        top++;
        stack[top]=val;
    }
}
int pop()
{
    int val;
    if(top==-1)
    {
        printf("stack underflow\n");
    }
    else
    {
        val=stack[top];
        top--;
    }
    return val;
}
```

Output:

enter prefix expression

+-234

RESULT:3

3. programme to evaluate postfix expression.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<math.h>
#define CAPACITY 100
int top=-1;
float stack[CAPACITY];
int get_type(char);
void push(float);
float pop();
void main()
{
    char postfix[100];
    float val,op1,op2,result;
    int i;
    printf("enter postfix expression\n");
    gets(postfix);
    for(i=0;postfix[i]!='\0';i++)
    {
```

```
        switch(get_type(postfix[i]))
        {
            case 0:
                val=(float)postfix[i]-'0';
                push(val);
                break;
            case 1:
                    op2=pop();
                    op1=pop();
                    switch(postfix[i])
                    {
                        case '+':
                                result=op1+op2;
                                break;
                        case '-':
                                result=op1-op2;
                                break;
                        case '*':
                                result=op1*op2;
                                break;
                        case '/':
                                result=op1/op2;
                                break;
                        case '%':
                                result=(int)op1%(int)op1;
                                break;
                        case '^':
                                result=pow(op1,op2);
                                break;
                    }
                    push(result);

        }
    }
    printf("result=%.2f",stack[0]);
}
int get_type(char c)
{
    if(c=='+'|| c=='-'||c=='*'||c=='/'||c=='%'||c=='^')
    {
        return 1;
    }
    else
    {
```

```
        return 0;
    }
}
void push(float val)
{
    if(top==CAPACITY-1)
    {
        printf("stack overow\n");
    }
    else
    {
        top++;
        stack[top]=val;
    }
}
float pop()
{
    float val;
    if(top==-1)
    {
        printf("underflow\n");

    }
    else
    {
        val=stack[top];
        top--;
    }
    return val;
}
```

Output:

```
enter postfix expression
23+42^-
result=-11.00
```

4. program to convert infix to postfix

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
#define CAPACITY 100
char stack[CAPACITY];
int top=-1;
void infix_to_postfix(char source[],char target[]);
```

```
int getpri(char);
void push(char stack[],char);
char pop(char stack[]);
void main()
{
    char infix[100],postfix[100];
    printf("enter infix expression\n");
    gets(infix);
    strcpy(postfix,"");
    infix_to_postfix(infix,postfix);
    puts(postfix);
}
void infix_to_postfix(char source[],char target[])
{
    int i=0,j=0;
    char temp;
    strcpy(target,"");
    while(source[i]!='\0')
    {
        if(source[i]=='(')
        {
            push(stack,source[i]);
            i++;
        }
        else if(source[i]==')')
        {
            while((top!=-1)&&(stack[top])!='(')
            {
                target[j]=pop(stack);
                j++;
            }
            if(top==-1)
            {
                printf("incorrect exp\n");
                exit(1);
            }
            temp=pop(stack);
            i++;
        }
        else if(isdigit(source[i])||isalpha(source[i]))
        {
            target[j]=source[i];
            j++;
            i++;
```

```
            }

    elseif(source[i]=='+'||source[i]=='-'||source[i]=='*'||source[i]=
='%'||source[i]=='/'||source[i]=='^')
            {

    while((top!=-1)&&(stack[top]!='(')&&getpri(stack[top])>=getpri(so
urce[i]))
                {
                    target[j]=pop(stack);
                    j++;
                }
                push(stack,source[i]);
                i++;
            }
            else
            {
                printf("incorrect exp\n");
                exit(1);
            }



        }
        while((top!=-1))
        {
            target[j]=pop(stack);
            j++;
        }
        target[j]='\0';

    }
    int getpri(char c)
    {
        if(c=='^')
        {
            return 2;
        }
        else if(c=='/'|| c=='*'||c=='%')
        {
            return 1;
        }
        else
        {
```

```
        return 0;
    }
}
void push(char stack[],char val)
{
    if(top==CAPACITY-1)
    {
        printf("overfloaw\n");
    }
    else
    {   top++;
        stack[top]=val;
    }
}
char pop(char stack[])
{
    return stack[top--];


}
```

Output:

```
enter infix expression
(a+b)/(c+d)-(d*e)
ab+cd+/de*-
```

5. program to convert infix to prefix

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
#define CAPACITY 100
char stack[CAPACITY];
int top=-1;
void reverse(char str[]);
void infixtopostfix(char source[],char target[]);
void push(char stack[],char);
char pop(char stack[]);
int getpri(char);
char infix[100],prefix[100],postfix[100];
void main()
{
    printf("enter infix expression\n");
    gets(infix);
```

```
    reverse(infix);
    strcpy(postfix,"");
    infixtopostfix(prefix,postfix);
    printf("corresponding postfix expression\n");
    puts(postfix);
    reverse(postfix);
    printf("prefix expression\n");
    puts(prefix);
}
void reverse(char str[])
{
    int len,i=0,j=0;
    len=strlen(str);
    for(i=len-1;i>=0;i--)
    {
        if(str[i]=='(')
        {
            prefix[j]=')';
        }
        else if(str[i]==')')
        {
            prefix[j]='(';

        }
        else
        {
            prefix[j]=str[i];
        }
        j++;

    }
    prefix[j]='\0';
}
void infixtopostfix(char source[],char target[])
{
    int i=0,j=0;
    char temp;
    strcpy(target,"");
    while(source[i]!='\0')
    {
        if(source[i]=='(')
        {
            push(stack,source[i]);
            i++;
```

```
            }
            else if(source[i]==')')
            {
                while((top!=-1)&&(stack[top]!='('))
                {
                    target[j]=pop(stack);
                    j++;
                }
                if(top==-1)
                {
                    printf("incorrect expression\n");
                    exit(1);
                }
                temp=pop(stack);
                i++;

            }
            else if(isdigit(source[i])||isalpha(source[i]))
            {
                target[j]=source[i];
                j++;
                i++;

            }

    elseif(source[i]=='+'||source[i]=='-'||source[i]=='*'||source[i]=
='/' ||source[i]=='%'||source[i]=='^')
            {

    while((top!=-1)&&(stack[top]!='(')&&(getpri(stack[top])>=getpri(s
ource[i])))
                {
                    target[j]=pop(stack);
                    j++;
                }
                push(stack,source[i]);
                i++;
            }
            else
            {
                printf("incorrect expression\n");
                exit(1);
            }
```

```
    }
    while((top!=-1))
        {
            target[j]=pop(stack);
            j++;
        }
        target[j]='\0';
}
void push(char stack[],char val)
{
    if(top==CAPACITY-1)
    {
        printf("stackoverflow\n");

    }
    else
    {
        top++;
        stack[top]=val;
    }
}
char pop(char stack[])
{
    char val=' ';
    if(top==-1)
    {
        printf("stack underlow\n");

    }
    else
    {
        val=stack[top];
        top--;
    }
    return val;
}
int getpri(char c)
{
    if(c=='^')
    {
        return 2;
    }
    else if(c=='*'||c=='/'|| c=='%')
    {
```

```
        return 1;
    }
    else
    {
        return 0;
    }
}
```

Output:

```
enter infix expression
x+(y*z/w)
corresponding postfix expression
wz/y*x+
prefix expression
+x*y/zw
```

6.) Program ro reverse the contents of stack

```
#include<stdio.h>
#include<conio.h>
#define CAPACITY 100
int top=-1;
int stack[CAPACITY];
void push(int);
int pop();
void main()
{
    int arr[100],n,i,val;
    printf("enter number of elements in array\n");
    scanf("%d",&n);
    printf("enter array elements\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    for(i=0;i<n;i++)
    {
        push(arr[i]);
    }
    for(i=0;i<n;i++)
    {
        val=pop();
        arr[i]=val;
    }
    printf("the reversed list\n");
    for(i=0;i<n;i++)
    {
```

```
            printf("%d\t",arr[i]);
        }

    }
    void push(int val)
    {
        top++;
        stack[top]=val;
    }
    int pop()
    {
        int val;
        val=stack[top];
        top--;
        return val;

    }
    Output:
    enter number of elements in array
    5
    enter array elements
    11
    12
    13
    14
    15
    the reversed list
    15      14      13      12      11
```

7.) Program to check nesting of parathesis

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define MAX 50
char stack[MAX];
int top=-1;
void push(char);
char pop();
void main()
{
    char exp[MAX],temp;
    int i,flag=1;
    printf("enter expression\n");
    gets(exp);
    for(i=0;i<strlen(exp);i++)
```

```
    {
        if(exp[i]=='('||exp[i]=='{'||exp[i]=='[')
        {
            push(exp[i]);
        }
        if(exp[i]==')'||exp[i]=='}'||exp[i]==']')
        {
            if(top==-1)
            {
                flag=0;
            }
            else
            {
                temp=pop();
                if(exp[i]==')'&&(temp=='['||temp=='{'))
                    flag=0;
                if(exp[i]=='}'&&(temp=='('||temp=='['))
                    flag=0;
                if(exp[i]==']'&&(temp=='('||temp=='{'))
                    flag=0;
            }
        }
    }
    if(top>=0)
        flag=0;
    if(flag==1)
    {
        printf("valid expression\n");
    }
    else
    {
        printf("invalid expression\n");
    }
}
void push(char c)
{
    if(top==MAX-1)
    {
        printf("stack overfloaw\n");
    }
    else
    {
        top++;
        stack[top]=c;
```

```
    }
}
char pop()
{
    char val;
    if(top==-1)
    {
        printf("stack underflow\n");

    }
    else
    {
        val=stack[top];
        top--;
        return val;
    }
}
```

Output:

```
enter expression
(a+{b*c})
valid expression
```