Objective

- Learning the Recursive Trace.

Q. # 1. Create a recursion trace (Recursion Tree) for the following algorithms using the provided starting value(s); also show the return value at each level of recursion.

a.

```
int f(int k, int n)
{
    if (n == k)
        return k;
    else if (n > k)
        return f(k,n-k);
    else
        return f(k-n,n);
}
Staring Values : k=6, n=8
```

b. Give recursive trace but also tell that what argument values, if any, could you pass to F that would cause the program to run forever?

```
int F(int N)
{
    cout<<"F entered with N = "<<N<<"\n";
    if (N >= 0 && N <= 2)
    {
        return N+1;
    }
    else
    {
        return F(N-2) * F(N-4);
    }
}
```

c.

```
int mystery(int x, int y)
{
    if (x < 0)
    {
        return -mystery(-x, y);
    }
    else if (y < 0)
    {
        return -mystery(x, -y);
    }
    else if (x == 0 && y == 0)
    {
        return 0;
    }
    else
    {
        return 100 * mystery(x / 10, y / 10) + 10 * (x % 10) + y % 10;
    }
}
```

Draw its recursive trace for following calls

- `mystery(7, -2);`
- `mystery(29, 45);`
- `mystery(135, 246);`

d.
```
int enigma ( int m, int n)
{
        if ( m==0 )
                return n+1;
        else if ( n==0 )
                return enigma(m-1,1);
        else
                return enigma( m-1, enigma( m, n-1) );
}
```
Starting values: m = 1 and n = 3.

e.
```
int mystery( int n )
{
        if ( n<=1 )
                return n;
        else if ( n%2 == 0 )
                return n + mystery(n/2);
        else return mystery( ( n+1)/2 ) + mystery( (n-1)/2 );
}
```
Starting value: n=13

f.
```
int oops( int n )
{
    int s=0;
       if ( n<=1 )
               return s;
    for ( int i=1; i<=n; i++ )
    {
         s = s + oops(n-i) + 1;
    }
    n=n-2;
    return s;
}
```
Starting value: n=4

Question given below are related to permutation and combination story: you may read the following to review these concepts: *http://www.mathsisfun.com/combinatorics/combinations-permutations.html*

Q. # 2. Consider the following function i.e. 'permutation', which display all the possible permutation of the given string. Your task is to show the recursive trace for string = "ABC"

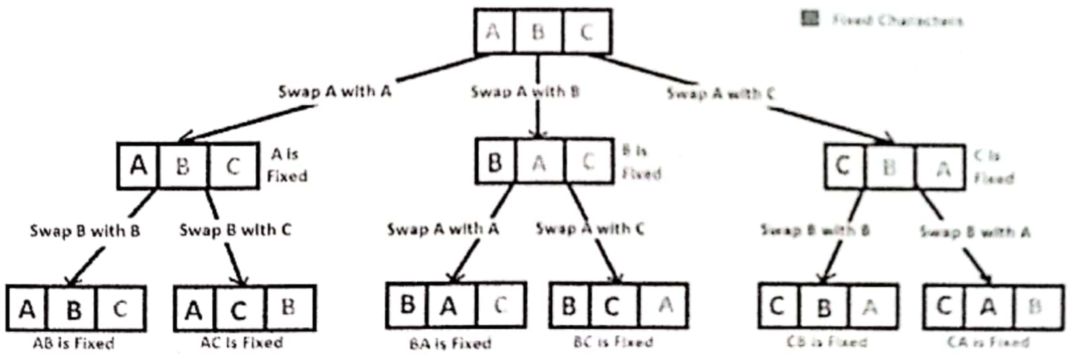
```
void swap(char *fir, char *sec)
{
    char temp = *fir;
    *fir = *sec;
    *sec = temp;
}
```

```
int main()
{
    char str[] = "ABC";

    permutation( str, 0, sizeof(str)-1 );
    return 0;
}
```

```
/* arr is the string, curr is the current index to start permutation from and size is
sizeof the arr */
void permutation(char * arr, int curr, int size)
{
    if(curr == size-1)
    {
        for(int a=0; a<size; a++)
            cout << arr[a] << "\t";
        cout << endl;
    }

    else
    {
        for(int i=curr; i<size; i++)
        {
            swap( &arr[curr], &arr[i] );
            permutation( arr, curr+1, size );
            swap( &arr[curr], &arr[i] );
        }
    }
}
```



Recursion Tree for Permutations of String "ABC"

Q. # 3. Trace the following program and see what it does.

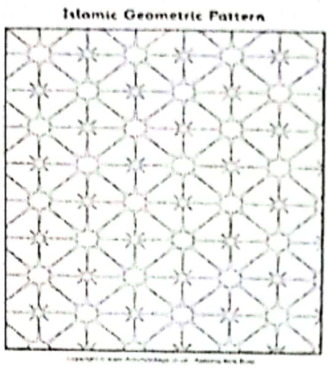
```
void gen(int *arr, int *temparr, int level, int
start, int N)
{
    int i, j;
    for (i=start; i<N; i++)
    {
        temparr[level] = arr[i];
        for (j=0; j<=level; j++)
            cout<<temparr[j]<<", ";
        cout<<endl;
        if( i < N-1)
```

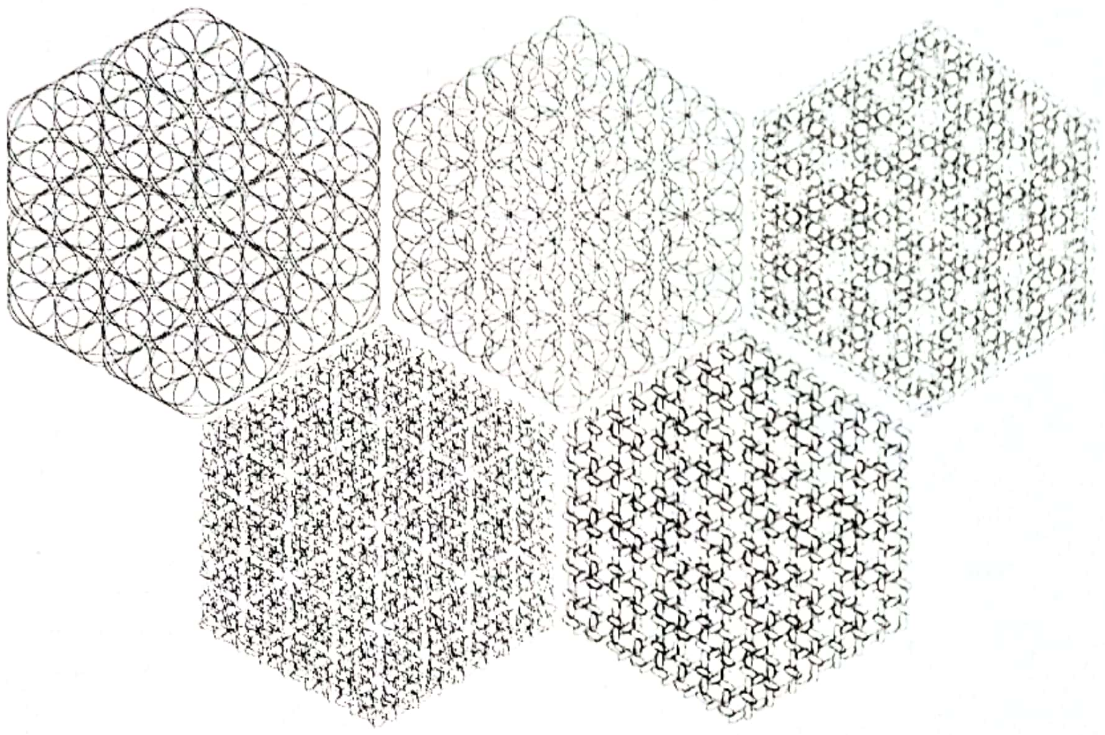
```
int main()
{
    int temparr[3];
    int a[3] = {1,2,3};
    gen(a,temparr, 0, 0,3);
    return 0;
}
```

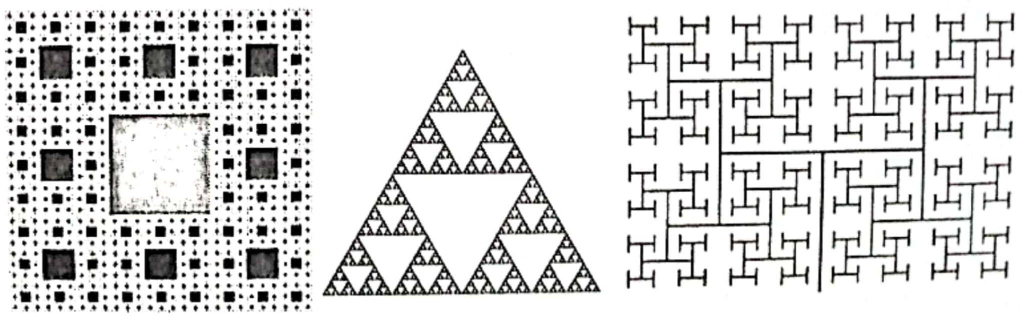
```
            gen(arr,temparr, level+1, i+1, N);
    }
}
```

Have you ever noticed the beautiful and stunning recursive fractals in Islamic architecture? ☺ Have a look at following link and may also see some amazing recursive designs/images taken from google.

https://en.wikipedia.org/wiki/Islamic_geometric_patterns







See this to see/do some fun: http://gregtatum.com/poems/recursive/3/