DSA ASSIGNMENT - 1

Q 19] Write a recursive function to count all the prime numbers between numbers a and b (both inclusive)

```
#include <stdio.h>
int primeNum( int num, int i ){
	if( i == num )
		return 1;
	else if( num % i == 0 )
		return  0;
	else
		primeNum( num, i+1 );
}

int countPrime( int a, int b ){
	if( a > b )
		return  0;
	return( primeNum( a, 2 ) + countPrime( ++a, b ) );
}
```

Q 21] A number is a perfect number if the sum of all it's positive proper divisors is equal to the number. Write a recursive function to check if a number is a perfect number or not.

```
#include <stdio.h>

int pFactors( int n, int i ){
	if( i == n )
		return 1;
	else if( n % i == 0 )
		return( i + pFactors( n, ++i) );
	else
		pFactors( n, ++i );
}

int main(){
	int a, p;
	printf( "Enter the number: " );
	scanf( "%d", &a );
	p=( pFactors( a, 2 ) );
	printf( "\n%d\n", p );
	if( p == a )
		printf( "\nIt is a perfect number." );
	else
		printf( "\nIt is not a perfect number." );
}
```

Vipul Chodankar, SE COMP
181105066, Batch D

Q 28] Write a recursive function to find the remainder when a positive integer a is divided by a positive integer b.

```
#include <stdio.h>

int rem( int a, int b ){
        if( b > a )
                return a;
        else
                rem( a - b , b );
}
```

Q 32] Write a recursive function to find the value of [log2N] and [lognN].

```
#include<stdio.h>

float logCal( float base, float num ){
        if( base / n == 1 )
                return 1;
        else
                return( 1 + logCal( base, num / base ) );
}
```

Q 34] Write a recursive function to compute Ackermann's functionA(m,n) which is defined as

```
A( m, n ) =     n + 1                   if m = 0
                A( m - 1, 1 )           if m > 0, n = 0
                A( m - 1, A( m, n – 1 ) )   otherwise
```

```
#include <stdio.h>

int cal( int m, int n ){
        if( m == 0 )
                return n + 1;
        else if( m > 0 && n == 0 )
                return cal( m – 1, 1 );
        else
                return cal( m – 1, cal( m, n - 1) );
}
```

Vipul Chodankar, SE COMP
181105066, Batch D

Q 40] Write a recursive function to check if a string is a palindrome or not.

```
#include <stdio.h>

int check( char a[], int i, int j ){
	if( i < j ){
		if( a[ i ] == a[ j - 1 ])
			return ( 0 + check( a, ++i, --j) );
	else
		return 1;
	}
	else
		return 0;
}
```

Q 42] Write a recursive function to convert a positive integer to string.

```
#include <stdio.h>

void printNumber( int n ){
	int k;
	if( n == 0 )
		return;
	k = n % 10;
	switch( k ){
		case 0: printNumber( n / 10 );
			printf( "  Zero" );
			break;

		case 1: printNumber( n / 10 );
			printf( "  One" );
			break;

		case 2: printNumber( n / 10 );
			printf( "  Two" );
			break;

		case 3: printNumber( n / 10 );
			printf( "  Three" );
			break;

		case 4: printNumber( n / 10 );
			printf( "  Four" );
			break;
```

```
        case 5: printNumber( n / 10 );
                printf( "  Five" );
                break;

        case 6: printNumber( n / 10 );
                printf( "  Six" );
                break;


        case 7: printNumber( n / 10 );
                printf( "  Seven" );
                break;

        case 8: printNumber( n / 10 );
                printf( "  Eight" );
                break;

        case 9: printNumber( n / 10 );
                printf( "  Nine" );
                break;
        }
}
```

Vipul Chodankar, SE COMP
181105066, Batch D