- Write Recursive functions for Linear Search in 2-D arrays using Recursion
- Write a C Program to Print Binary Equivalent of an Integer using Recursion
- Write a function to find the Biggest Number in an Array of Numbers using Recursion
- Write a function to perform Matrix Multiplication using Recursion
- Write a function to everse the String using Recursion
- Write a function to find reverse of a number using Recursion
- Write a function to copy one string to another using Recursior
- Write a function to find Least Common Multiple (LCM) of a given number using Recursion
- Write a function to convert a Number Decimal System to Binary System using Recursion
- Write a function to find the first capital Letter in a string using Recursion

- Write a recursive function to print the values of a 1D-array of integers
- Power Set: Write a recursive program to generate all subsets of a given string (Given a set represented as a string of characters)
- Write a recursive program to check whether a given number is palindrome or not?
- Write a recursive program to check whether a given number of prime or not?
- Write a recursive program to return the sum of the first N natural numbers
- Given a binary number as string, write a recursive program to find its decimal equivalent.
- Assume sufficiently a long integer. Write a recursive program to sum up all prime digits of that number.
- Write a Recursive function that prints all numbers less than N which consist of digits, each can be of 1 or 3 or multiples of 3 (if N = 20, then output is 19, 16, 13, 11, 9, 6, 3, 1)

```c
#include <stdio.h>
int search(int arr[][4],int i, int j, int ele,int row,int col)
{
    if(i>row)
    return -1;
    if(j>col)
    return search(arr,i+1,0,ele,row,col);
    if(arr[i][j]==ele)
    return 1;
    return search(arr,i,j+1,ele,row,col);

}
int main()
{
    int arr[3][4];
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<4;j++)
        {
            scanf("%d",&arr[i][j]);
        }
    }
int ans=search(arr,0,0,67,3,4);
printf("%d \n",ans);
return 0;
}
```

very important linear search in 2D array.

```c
#include <stdio.h>
int decToBi(int n)
{
    if(n!=0)
    return decToBi(n/2)*10+n%2;
    else
    return 0;
}
int main()
{
int ans=decToBi(11);
printf("%d",ans);
return 0;
}
```

decToBi(11) = decToBi(5)*10 + 1 → 101
decToBi(2)*10 + 1 → 10
decToBi(1)*10 + 0
decToBi(0)*10 + 1
0
1
fn return 1011
8421
= 11

return decToBi(n/2)*10 + n/2

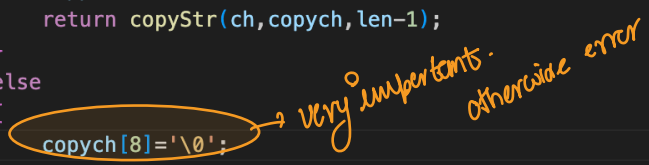
```c
#include <stdio.h>
int maxEle(int arr[],int max,int len)
{
    if(len<0)
    return max;
    if(arr[len]>max)
    max=arr[len];
    return maxEle(arr,max,len-1);
}
int main()
{
int arr[]={1,2,3,4,5,23,6,7,8};
int ans=maxEle(arr,0,8);
printf("%d\n",ans);
return 0;
}
```

```c
#include <stdio.h>
char* reverse(char str[],int len,int start)
{
    if(start>len)
    return str;
    else
    {
        char ch;
        ch=str[start];
        str[start]=str[len];
        str[len]=ch;
        return reverse(str,len-1,start+1);
    }
}
int main()
{
char arr[]="sarthak";
char* rev=reverse(arr,6,0);
printf("%s\n",rev);
return 0;
}
```

```c
#include <stdio.h>
char* reverse(char str[],int len,int start)
{
    if(start>len)
    return str;
    else
    {
        char ch;
        ch=str[start];
        str[start]=str[len];
        str[len]=ch;
        return reverse(str,len-1,start+1);
    }
}
int main()
{
char arr[]="sarthak";
char* rev=reverse(arr,6,0);
printf("%s\n",rev);
return 0;
}
```

```c
#include <stdio.h>
char* copyStr(char ch[],char copych[],int len)
{
    if(len>=0)
    {
        copych[len]=ch[len];
        return copyStr(ch,copych,len-1);
    }
    else
    {
        copych[8]='\0';
    return copych;
    }

}
int main()
{
    char ch[]="My World";
    char copych[8];
char* copy=copyStr(ch,copych,7);
printf("%s",copy);
return 0;
}
```



```c
#include <stdio.h>
int gcd(int a,int b)
{
    if(b!=0)
    {
        int n=a%b;
    return (b,n);
    }
    else
    return a;

}
int main()
{
int ans=(6*9)/gcd(9,6);
printf("%d",ans);
return 0;
}
```

return gcd(b, a%b)

m > n

Recursion Exercieses > C capletter.c > main()

```c
#include <stdio.h>
int cap(char ch[],int len ,int ind)
{
    if(ind<len)
    {
        if((int)ch[ind]>=65 && (int)ch[ind]<=90)
        return ind;
        return cap(ch,len,ind+1);
    }
    else
    return -1;
}
int main()
{
    char arr[]="sarthAk";
int ch=cap(arr,7,0);
printf("%d",ch);
return 0;
}
```

```c
#include <stdio.h>
void print(int arr[],int ind,int len)
{
    if(ind<len)
    {
    printf("%d\n",arr[ind++]);
    print(arr,ind,len);
    }
    else
    return;
}
int main()
{
int arr[]={1,2,3,4,5,6};
print(arr,0,6);
return 0;
}
```

int [6]{(int)1, (int)2, (int)3, (int)4, (int)5, (int)6}

```c
#include <stdio.h>
int reverse(int num,int rev)
{
    if(num!=0)
    {
        rev=rev*10+num%10;
        return reverse(num/10,rev);
    }
    else
    {
        return rev;
    }
}
int main()
{
int num=16551;
if(num==reverse(num,0))
printf("it is a palindrome nuumber\n");
else
printf("it is not a palndrome number\n");
return 0;
}
```

```c
#include <stdio.h>
#include <stdbool.h>
bool prime(int num,int i)
{
    if(i==num)
    return true;
    else
    {
        if(num%i==0)
        return false;
        else
        return prime(num,i+1);
    }
}
int main()
{
if(prime(18,2))
printf("it is a prime number\n");
else
printf("not a prime number\n");
return 0;
}
```

important

```c
#include <stdio.h>
#include <math.h>
int dec=0;
void biToDec(char* ch,int ind,int len)
{
if(ind<=len)
{
    if(ch[len-ind]=='1')
    dec+=1*pow(2,ind);
    else
    dec+=0;
    biToDec(ch,ind+1,len);
}
else
return;
}
int main()
{
    char ch[]="101101";
    int len=sizeof(ch)/sizeof(ch[0]);
    biToDec(ch,0,5);
printf("%d \n",dec);
return 0;
}
```

important

1 0 1 1
3 2 1 0

$2^3 + 2^1$

16 8 4 2 1

```c
#include <stdio.h>
#include <stdbool.h>
bool checkPrime(int n,int i)
{
    if(n==i)
    return true;
    else
    {
        if(n%i==0)
        return false;
        else
        return checkPrime(n,++i);
    }
}
int sumDig(int n)
{
    if(n!=0)
    {
        int dig=n%10;
        if(checkPrime(dig,2))
        return dig+sumDig(n/10);
        else
        return sumDig(n/10);
    }
    else
    return 0;
}
int main()
{
int num=678935;
int ans=sumDig(num);
printf("%d\n",ans);
return 0;
}
```

Recursion Exercises > C PRINT.c > main()

```c
#include <stdio.h>
#include <stdbool.h>
bool check(int dig)
{
    if(dig==1 || dig%3==0)
        return true;
    return false;
}
bool checkNum(int num)
{
    if(num!=0)
    {
        int dig=num%10;
        if(check(dig))
        return checkNum(num/10);
        else
        return false;
    }
    else
    return true;
}
void printN(int N,int i)
{
    if(i<N)
    {
        if(checkNum(i))
        printf("%d ",i);
        printN(N,i+1);
    }
    else
    return;
}
int main()
{
printN(20,1);
return 0;
}
```

+ combinations. + pointers

```
const int MAX = 100;

void multiplyMatrixRec(int row1, int col1, int A[][MAX],
                       int row2, int col2, int B[][MAX],
                       int C[][MAX])
{
    // Note that below variables are static
    // i and j are used to know current cell of
    // result matrix C[][]. k is used to know
    // current column number of A[][] and row
    // number of B[][] to be multiplied
    static int i = 0, j = 0, k = 0;

    // If all rows traversed.
    if (i >= row1)
        return;

    // If i < row1
    if (j < col2) {
        if (k < col1) {
            C[i][j] += A[i][k] * B[k][j];
            k++;

            multiplyMatrixRec(row1, col1, A, row2, col2, B,
                              C);
        }

        k = 0;
        j++;
        multiplyMatrixRec(row1, col1, A, row2, col2, B, C);
    }

    j = 0;
    i++;
    multiplyMatrixRec(row1, col1, A, row2, col2, B, C);
}
```