Q Write a program to convert binary to decimal & decimal to binary

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

{
int flag = 0;
int inp;

do {
printf("If you want to convert Decimal number to binary number, type 1. & If you want to convert Binary to Decimal, type 2\n");
```

```
scanf("%d", &inp);
if (inp == 1)
{
    flag = 1;
    int n, bi=0, i=1;
    printf("Enter a number: ");
    scanf("%d", &n);

    printf("%d in binary is ", n);
    while (n!=0)
    {
        bi = bi + ((n%2) * i);
        n = n/2;
        i = i*10;
    }
    printf("%d", bi);
}

else if (inp == 2)
{
    flag = 1;
int bi, n, temp, sum=0;
printf("Enter the binary number: \n");
scanf("%d", &bi);
    n = bi;
for (int i=0 ; n>0 ; i++)
{
temp = (n - ((n/10) * 10)) * pow(2, i);
    n = n/10;
    sum = sum + temp;
}
```

```
printf("Decimal of %d in binary is
                %d\n", li, sum);
    }
    else
    {
    flag=0;
    printf("Wrong choice\n");
    }
    } while(flag==0);
      return 0;
      }
```

Output

If you want to convert - - -
- - - - type 2

1

Enter a number: 928
928 in binary is 1110100000

Q Twin primes are consecutive odd nos, both of which are prime nos. Write a program which inputs 2 positive nos A & B and outputs all twin primes in range A to B

```
#include <stdio.h>
int main()
{
int n1, n2, i, j;
int prime1, prime2, prmchk=1;
printf("Enter starting number: ");
scanf("%d", &n1);
printf("Enter ending number: ");
scanf("%d", &n2);
```

```
for(i=n1; i<=n2; i++)
{
  for(j=i-1; j>1; j--)
    if(i%j==0)
    {
      prmchk=0;
      break;
    }
  if(prmchk==1)
  {
    preme1=i;
    if(preme1-preme2==2)
    {
      printf("(%d,%d)", prime1, preme2);
    }
    preme2=preme1;
    prmchk=1;
  }
  return 0;
}
```

Output: Enter starting number: 7
Enter ending number: 129

(13,11), (19,17), (31,29), (43,41) ..... (109,107).

Q Write a prog. to find out whether a ng. is kaprekar or not. Consider an n-digit ng. k.
Sq. it & add the right n digits to the left n or n-1 digits. If the resultant sum is k, then

Date
Page

It is called Kaprekar ng.. for. $9 \rightarrow 9^2 = 81 \Rightarrow 8+1=9$

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

int main()
{
int n, k1, k2, sq, digits, num, power;
printf("Enter a number: ");
scanf("%d", &n);
sq = n*n;
while (sq != 0)
{
 digits ++;
 sq = sq / 10;
}
if (digits %2==0){}
else{
   digits = digits +1;
}
 sq = n*n;
num = digits/2;
power = pow(10, num);
k1 = sq % power;
k2 = (sq - k1)/power;
if (k1+k2 == n){
 printf("It is a Kaprekar number");
}
else{ printf("It not a Kaprekar ng."); }

 return 0;
}
```

Output: Enter a number: 703
It's a Kaprekar no.

Q Write a C program that calculates the total income on amount Rs. 5 lakhs in a period of 10yrs. Show the results for simple interest, compounded interest when the compounding is done annually, semi-annually, quarterly, monthly, daily. Assume that the interest rate is 3.5% per year.

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

int main()
{
  float p,t,r;

printf("Enter principal, time period, rate:\n");
scanf("%f %f %f",&p,&t,&r);
float si;
si=p*t*r;
printf("Simple interest:%f\n",si);
int N[5]={1,2,4,12,365};
int ctr=0;

while(ctr<=4)
{ float c=pow(1+(r/N[ctr]),N[ctr]*t);
float ci=p*c-p;
printf("Compound interest compounded %d
  times a year=%f)\n",N[ctr],ci);
  ctr++; }
```

return 0;
}

Output
Enter principal, time period, rate:
500000
10
0.035
Simple Interest: 175000.000

Compound int: compounded 1 times a year = 2052991.2
" " 2 " = 2073889.68
" " 4 " = 2084533.37
" " 12 " = 2091746.8
" " 365 " = 2094022.37

Array: collection of data items of same type, that are associated with a particular symbolic name.

~~Declare both the no. of~~

Q Given a number n, print following pattern without using any loop. E.g. n=16 Output 16, 11, 6, 1, -4, 1, 6, 11, 16. E.g. n=10 Output 10, 5, 0, 5, 10

Code:
```
#include <stdio.h>
int series(int);
int main()
{ int i, n;
printf("Enter a number: \n");
```

```
scanf("%d", &n);
series(n);

return 0;
}
int series(int n)
{
int num;
if(n<=0)
{
printf("%d", n);
return n;
}

else
{
printf("%d", n);
num = 5+ series(n-5);
printf("%d", num);
return num;
}
}
```

Output

```
Enter a number:
16
16 11 6 1 -4 1 6 11 16
```

If There are atleast two sequences attributed to B. Recaman. One is the sequence an formed by taking $a_1 = 1$ and letting, ~~$a_n = a_n - 1$~~

$a_n = a_{n-1} - n$ if $a_{n-1} - n > 0$ & is new

$= a_{n-1} + n$ otherwise

which can be succinctly defined as "subtract if you can, otherwise add. The first few terms are

1, 3, 6, 2, 7, 13, 20, 12, 21, 11 .... so on

```
#include<stdio.h>
int rec(int n)
{
  if(n==1)
  {
    return 1;
  }
  else if((rec(n-1)-n)>0)
  {
    return rec(n-1)-n;
  }
  else
  {
    return rec(n-1)+n;
  }
}
int main()
{
int n;
printf("Enter a number: ");
scanf("%d",&n);
printf("Recaman series upto the %dth term
                    is:\n",n);
```

```
for(int i=1; i<=n; i++){
    printf("%d", rec(i));
}
return 0;
}
```

Output: Enter a number: 15

Recaman series upto the 15th term is:
1 3 6 2 7 1 8 16 7 17 6 18 5 19 4

Q Ackermann's function is defined by: ~~A(m,n)=n+1 if m=0~~

$A(m,n) = n+1$ if $m=0$
$= A(m-1,1)$ if $m\neq 0$ & $n=0$
$= A(m-1, A(m,n-1))$ if $m\neq 0$ & $n\neq 0$

Write a function which ~~given~~ given m & n returns A(m,n). Calculate the values of A(m,n) for all m in the range 1 to 3 & all n in the range.

Code:
```
#include <stdio.h>
int ack(int m, int n){
if(m==0)
{return n+1;}
else if(m!=0 && n==0)
{return ack(m-1,1);}
else if(m!=0 && n!=0)
{return ack(m-1, ack(m,n-1));
}}
```

```
void table()
{
int m = 1;
int n = 1;
printf("m | n | A(m,n) \n");
printf("                \n");
while (n<=6)
  {
   while (m<=3)
    {
     printf("%d | %d | %d \n", m, n,
                        ack(m,n));
                m++;
    }
    m=1;
    n++;
  }
}
int main()
{
  table();
  return 0;
}
```

Output

```
m | n | A(m,n)
```

1	1	3
2	1	5
3	1	13
1	2	4
2	2	7
3	2	29
1	3	5
2	3	9
3	3	61
1	4	6
2	4	11
3	4	125
1	5	7
2	5	13
3	5	253
1	6	8
2	6	15
3	6	509

Bubblesort

```
#include <stdio.h>

void bubblesort(int arr[], int size)
{
  int i, j;
  for (i=0; i<size; i++)
  {
    for(j=0; j<size-1; j++) // last element would be automatically sorted
    {
      if(arr[j]>arr[j+1])
        swap(&arr[j], &arr[j+1]
    }
```

```
}
}
void swap(int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

int main()
{
   int array[100], i, size;
printf("How many nos you want to sort:");
scanf("%d", &size);
printf("\n Enter %d numbers:", size);

for(i=0; i<size; i++){
      scanf("%d", &array[i]);
bubblesort(array, size);
printf("\n sorted array is");
   }
  for(i=0; i<size; i++){
  printf("%d", array[i]);
   return 0;
    }
```

Output

How many nos you want to sort:5

Enter 5 nos: 345 3 20 35 333

Sorted array is : 3 20 35 333 345

~~Insertion~~

Insertion Sort

1) We will store the random set of nos in an array

2) We will traverse the array & insert each element of this array, to its correct posn where it should actually be, by shifting the other elements if reqd.

3) The 1st element in the array is considered as sorted, even if it is an unsorted array. The array is subdivided into 2 parts, the 1st part holds the 1st element of the array which is considered to be sorted & second part contains all the rem. elements of array.

4) With each iteration one element from the second part is picked & inserted into the 1st part of array at its correct posn by shifting existing elements if reqd.

5) This goes until last element in second part of array is placed in correct posn.

```
#include <stdio.h>
int main()
{
int n, i, j, temp;
int arr[64];

printf("Enter no. of elements\n");
scanf("%d", &n);

printf("Enter %d integers\n", n);
for(i=0; i<n; i++)
{
  scanf("%d", &arr[i]);
}
for(i=1; i<=n-1; i++)
{
  j=i;
  while(j>0 && arr[j-1]>arr[j])
  {
    temp = arr[j];
    arr[j] = arr[j-1];
    arr[j-1]= temp;
    j--;
  }
}
printf("Sorted list in ascending order:\n");

for(i=0; i<=n-1; i++)
{
  printf("%d\n", arr[i]);
}
return 0;
}
```

Enter no. of elements
6

Enter 6 integers
4 6 1 2 5 3

Sorted list in ascending order:
1
2
3
4
5
6

Selection Sort

1) Starting from the beginning pick one no.
2) Compare it with others one by one.
3) Replace if other no. is lesser than this one.
4) Display result.

```
#include <stdio.h>
void selection sort
void selection sort (int arr[], int size);
void swap (int *a, int *b);

void selection sort (int arr[], int size)
{
  int i, j;
```

```
for (i=0; i<size; i++)
{
for (j=1; j<size; j++)
{
if (arr[i] > arr[j])
    swap (&arr[i], &arr[j]);

}
}
}

void swap (int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

int main ()
{
int array[10], i, size;
printf("How many nos you want to sort:");
scanf ("%d", &size);

for(i=0; i<size; i++)
    scanf("%d", &array[i]);
selection_sort(array, size);
printf("\n sorted array is");
for (i=0; i<size; i++)
    printf("%d", array[i]);
return 0;
}
```

How many nos you want to sort: 3

Enter 3 nos:
-3
31
66

sorted arrays: -3 31 66

Quick Sort

```
#include <stdio.h>

void quicksort(int [], int, int);

int main()
{
int list[50];
int size, i;

printf("Enter the number of elements: ");
scanf("%d", &size);
printf("Enter the elements to be sorted:\n");

for(i=0; i<size; i++)
{
  scanf("%d", &list[i]);
}
quicksort(list, 0, size-1);
```

```
printf("After applying Quicksort:\n");
for(i=0; i<size; i++)
{
  printf("%d", list[i]);
}
printf("\n");

  return 0;
}

void quicksort(int list[], int low,
               int high)
{ int pivot, i, j, temp;
  if(low < high)
  {
    pivot = low;
    i = low;
    j = high;
    while(i<j)
    {
while(list[i] <= list[pivot] && i<=high)
    {
      i++;
    }
while(list[j] > list[pivot && j>=low)
    {
      j--;
    }
    if(i<j)
    {
      temp = list[i];
```

```
        list[i] = list[j];
        list[j] = temp;
      }
    }
    temp = list[j];
    list[j] = list[pivot];
    list[pivot] = temp;
    quicksort(list, low, j-1);
    quicksort(list, j+1, high);
  }
}
```

Output:

```
Enter the no. of elements: 6
Enter the elements to be sorted:
67
45
24
98
12
38

After applying quicksort

12 24 38 45 67 98
```

Linear Search

```
#include <stdio.h>

void main()
{
  int num;
```

```
int i, keynum, found=0;
printf("Enter the no. of elements");
scanf("%d", &num);
int array[num];
printf("Enter the elements one by one");
for (i=0; i<num; i++)
{
  scanf("%d", &array[i]);
}

printf("Enter the element to be searched");
scanf("%d", &keynum);

// Linear search begins
for(i=0; i<num; i++)
{
  if(keynum == array[i])
  {
    found=1;
    break;
  }
}
if(found==1)
  printf("Element present in the
          array at posn %d", i+1);
```

```
else
    printf("Element is not present in the
                    array \n");
```

Output:

```
Enter the no. of elements: 6
Enter the elements one by one
4
6
1
2
5
3

Enter the element to be searched 6
Element is present in the array at
pos 2
```

BINARY SEARCH

```
#include <stdio.h>

void binary_search(int [], int lo, int hi,
                   int key);

void bubblesort(int list[], int size);

int main()
{
  int key, size, i;
  int list[25];

  printf("Enter size of list: ");
  scanf("%d", &size);
  printf("Enter elements \n");
    for(i=0; i<size; i++){
      scanf("%d", &list[i]);
    }

  bubble_sort(list, size);

  printf("Enter key to search\n");
  scanf("%d", &key);
  binary_search(list, 0, size, key);
}

void bubble_sort(int list[], int size)
{
    int temp, i, j;
```

```
for(i=0; i<size; i++)
{
 for(j=i; j<size; j++)
 {
  if (list[i] > list[j])
  {
     temp = list[i];
     list[i]=list[j];
     list[j]=temp;
  }
 }
}

void binary_search(int list[], int lo, int hi,
                                    int key)
{
int mid;

if(lo>hi)
{
  printf("Key not found\n");
     return;
}
mid=(lo+hi)/2;
if (list[mid] == key)
{
  printf("Key found\n");
}
else if(list[mid] >key)
{
  binary_search(list, lo, mid-1, key);
}
```

```
else if (list[mid]<key)
{
  binary_search(list, mid+1, hi, key);
}
}
```

Output:

```
Enter size of list: 6
Enter elements
1
2
3
4
5
6
Enter key 6
Key found
```

Alternate algo for bin search

```
void bin_srch(int arr[80], int size, int num)
{
int hi=size-1; lo=0, i;

for(i=(size/2); arr[i]!=num; i=((hi+lo)/2))
{
 if(arr[i]<num){lo=i+1;}
else if (arr[i]>num){hi=i-1;}
if (i==0 || i==size-1){ break;}
}
 if(arr[i]=num) {printf("arr[%d]=%d", i,
                                arr[i]);}
```

else {printf("Element doesn't exist on the array");}
}

ARRAY Deleting REVERSAL

1) Create an array of some size, and fill up its elements.
2) Take a value as input from users, which needs to be deleted
3) Using for loop, check whether the value is present on the array or not.
4) If the value is present, then save its location & if it's not present, some appropriate message gets printed.
5) Again the loop runs from that saved pos^n till end of array, causing each element to shift by one step.

```
#include <stdio.h>
void main()
{
int vector[10];
int i, n, pos, element, found=0;

printf("Enter how many elements \n");
  scanf("%d", &n);
```

```
printf("Enter the elements \n");
for(i=0; i<n; i++)
{
  scanf("%d", &vectorx[i]);
}

printf("Enter the element to be deleted \n");
scanf("%d", &element);

for(i=0; i<n; i++)
{
  if (vectorx[i]==element)
  {
    found=1;
    pos=i;
    break;
  }
}
if(found==1)
{
  for (i=pos; i<n-1; i++)
  {
    vectorx[i]=vectorx[i+1];
  }
printf("The resultant vector is \n");
 for(i=0; i<n-1; i++)
 {
  printf("%d\n", vectorx[i]);
  }
}
else printf("Element not found \n
```

```
else {
  printf("Element %d is not found in
          the vector \n", element);
}
```

```
Enter how many elements
 4
Enter the elements
 345
 234
 678
 987

Enter the element to deleted
 234

The resultant vector is
 345
 678
 987
```

FLOYD'S TRIANGLE

```
#include <stdio.h>

main()
{
int i, j, k=1;

printf("Floyd's triangle is \n");
for(i=1; k<=20; ++i){
```

```
for(j=1; j<=i; ++j)
    printf("%d", k++);
    printf("\n\n");
}
return 0;
}
```

Floyd's triangle

1

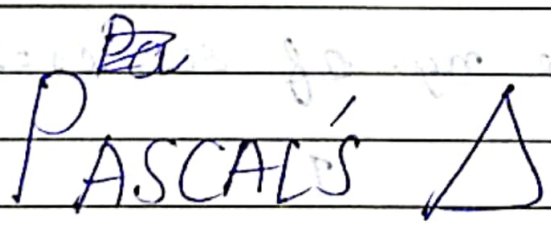
2 3

4 5 6

7 8 9 10

11 12 13 14 15

16 17 18 19 20 21




```
#include <stdio.h>
void main()
{
int array[15][15], i, j, rows, num=25, k;
printf("\n Enter the no. of rows: ");
scanf("%d", &rows);
```

```
for(i=0; i<rows; i++)
{
for(k=num-2*i; k>=0; k--)
    printf(" ");
for(j=0; j<=i; j++)
{
if(j==0 || i==j)
{
  array[i][j]=1;
}
else
{
array[i][j]=array[i-1][j-1]+array[i-1][j];
}
printf("%4d", array[i][j]);
}
printf("\n");
}
```

Output:

Enter the no. of rows: 2

```
   1
  1 1
```

STRING REVERSAL

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

```
void main()
{
int i, j=0, k=0, x, len;
char str[100], str1[10][20], temp;

printf("Enter the streng:");
scanf("%[^\n]s", str);

/* reads into 2-D character assay */

for (i=0; str[i]!='\0'; i++)
  {
   if (str[i]==' ')
    {
     str1[k][j]='\0';
      k++;
      j=0;
    }
   else
    {
    str1[k][j]= str[i];
      j++;
    }
}
str1[k][j]='\0';

/* reverses each word of a given streng*/
for(i=0; i<=k; i++)
  {
   len = strlen(str1[i]);
   for(j=0; x=len-1; j<x; j++, x--)
```

```
{
temp = str[i][j];
str1[i][j] = str1[i][x];
str1[i][x] = temp;
}
}

for(i=0; i<=k; i++)
{
printf("%s", str1[i]);
}
}
```

Output:

Enter the string: C Programming class

C gnimmargorP ssalC

Exp (1) Take a string as input & store it in the array str[].

(2) Using for loop store each word of the input string into the 2-D array str1[][].

(3) In the 2-D array str1[][] reverse each word of the string at each row of the array.

(4) Print the 2-D array as output.

DEL. REPEATED CHARACTERS

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

int main()
{
char str[100], word[100], twoD[10][30];
int i=0, j=0, k=0, len1=0, len2=0, l=0;

printf("Enter the string\n");
gets(str);

// String input into 2-D array

for(i=0; str[i]!='\0'; i++)
{
   if(str[i]==' ')
   {
      twoD[k][j]='\0';
      k++;
      j=0;
   }

   else
   {
      twoD[k][j]=str[i];
      j++;
   }
}
~~twoD[k]~~ twoD[k][j]='\0';
      j=0;
```

```
for(i=0; i<k; i++)
  {
  int present=0;
for(l=1; l<k+1; l++)
{
  if(twoD[l][j]=='\0' || l==i)
  {
    continue;
  }
if(strcmp(twoD[j], twoD[l])==0){
    twoD[l][j]='\0';
    present=present+1;
  }
}
/* if(present>0)
  {
  twoD[i][j]='\0';
  }
```

Uncomment this block if you want to remove all occurrences including the word itself

```
}
j=0;

for(i=0; i<k+1; i++)
  {
  if(twoD[i][j]=='\0')
        continue;
  else
        printf("%s", twoD[i]);
}
printf("\n");
return 0;
```

Output:

→ Welcome to Adwaith C prog. class,
welcome again to C class!

→ Welcome to Adwaith C prog. class, again!

C Arrays MCQ's

(1) Predict output

```
int main()
{
 int arr[5];
 // Assume that the base address of arr is 2000
// & size of integer is 32 bit

arr++;
printf("%u", arr);

return 0;
}
```

(A) 2002 (B) 2004 (C) 2020 (D) Value reqd.

Array name in C is implemented by a const. pointer. It is not possible to apply increment & decrement on const. types

2.

```
int main()
{
  int arr[5];
// Ass: base address of arr is 2000
// & size of integer is 32 bit
printf("%u %u", arr+1, &arr+1);

return 0;
}
```

(A) 2004 2020 (B) 2004 2004

(C) 2004 Garbage value (D) The prog. fails to compile because Address of operator cannot be used with array name

Name of array in C gives the address (except in size of operator) of the 1st element. Adding 1 to this address gives the address plus the size of type the array has. Applying the address of operator before the array name gives the address of the whole array. Adding 1 to this address gives the address plus the size of whole array

(2) void print(int arr[])
{
}

Q3 What would be the output of the program?

```
#include<stdio.h>

main()
{
int a[5]={5,1,15,20,25};
int i,j,m;

i=++a[1];
j=a[1]++;
m=a[i++];
printf("%d, %d, %d", i, j, m);
}
```

Ans: → int a[5]={5,1,15,20,25}. The variable arr is declared as an integer array with a size of 5 & it is initialized to a[0]=5,

→ i=++a[1], becomes i=++1; Hence i=2 & a[1]=2

→ j=a[1]++; becomes j=2++; Hence j=2 & a[1]=3

→ m=a[i++]; becomes m=a[2]; Hence m is 15 & i is incremented by 1 (i++ means 2++ so i=3)

∴ Output is 3, 2, 15

Q Output?

```
main()
{
char p;
char buf[10]={1,2,3,4,5,6,9,8};
p=(buf+1)[5];
printf("%d",p);
return 0;
}
```

Ans: &x[i] is equivalent to *(x+i)
so (buf+1)[5] is *(buf+1+5) i.e. buf[6]

Array elements are stored in sequential memory locations.

Q Let x be an array. Which of the foll. operations are illegal?

I ++x II x+1 III x++ IV x*2	S+ I, III & IV are invalid S+ I & III: ++x & x++ are throwing an error while compiling (lvalue req as increment operand) Since, x is storing the address of the array which is static value which cannot be changed by operand

St IV: x*2 would also throw an error while compiling (invalid operands to binary* (have int* & int)

St III: x+1 is throw a warning: assignment makes integer from pointer without a cast [enabled by default]

→ Size of an array need not be specified when it's a Declaration, formal parameter, Part of definition.

Q typedef char x[10];

x myarray[5];

what will sizeof(my array) be? (Assume one character occupies 1 byte)

Ans 50

Q Output?

main()

{

char str1[] = "abcd";
char str2[] = "abcd";

if (str1 == str2)
printf("Equal");
else
printf("Unequal");

}

When strings are compared using assignment operator their addresses are compared & as addresses are never equal, it prints Unequal.

Q Output?

```
main()
{
int a[10];
printf("%d %d", a[-1], a[12]);
}
```

Ans: Garbage value Garbage value
In c compiler doesn't check array with its bounds, value at the computed ~~the~~ location is displayed.

Q What does the declaration mean
int (*ptr)[10];

ptr is a pointer to an array of 10 integers.

→ Array passed as an argument to a function is interpreted as Address of the 1st element of the array.

Q What will be the output of the program if the array begins at 65472 & each integer occupies 2 bytes)

```
main()
{
    int a[3][4]={1,2,3,4,4,3,2,1,7,8,9,0};
    printf("%u, %u", a+1, &a+1);
}
```

Ans: The base address (also the address of the 1st element) of array is 65472.

For a 2D array like a reference to array has type "pointer to array of 4 ints". Therefore a+1 is pointing to the memory location of the 1st element of the 2nd row in array a. Hence 65472 + (4 ints * 2 bytes) = 65480

Then, &a has type "pointer to array of 3 arrays of 4 ints", totally 12 ints. ∴ &a+1 denotes "12 ints * 2 bytes * 1 = 24 bytes"

Hence, beginning address 65472 + 24 = 65496.
∴ &a+1 = 65496

---

arr[i] = i[arr]

→ In a 1D array a[100], the element a[i] can be accessed as a[i] or *(a+i) or *(i+a)

→ Address of a[i] can be accessed as &a[i] or (a+i) or (i+a)

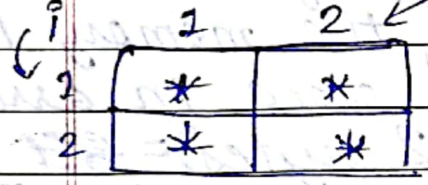
→ In a 2D array a[100][100], the

element a[i][j] can be accessed as a[i][j] or *(*(a+i)+j) or *(a[i]+j)

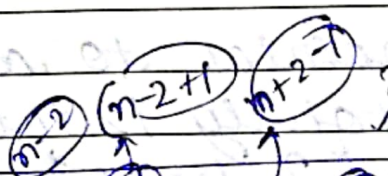
→ Address of a[i][j] can be accessed as &a[i][j] or a[i]+j or *(a+i)+j

→ In a 2D array, address of i^{th} row can be accessed as a[i] or *(a+i)

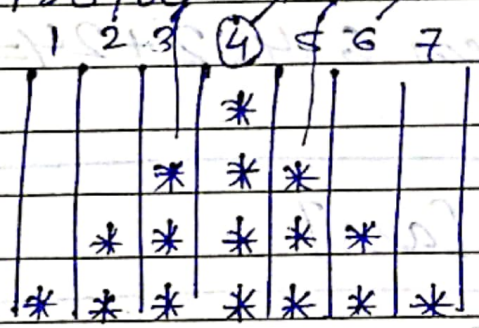
Patterns




```
for(i=1; i<=2; i++)
{
for(j=1; j<=2; j++)
{
printf("*");
}
}
```



Right



If 3 rows → 5 col
4 row → 7 col
5 row → 9 col
6 row → 11 col
n row → 2n-1 col

Draw a matrix
Find Relation bet^n rows & col.
Find rel^n bet^n rows & col: for printed things (stars here)

```
for (i=1; i<=n; i++)
{
for(j=1; j<=2*n-1; j++)
  {
  if(j>=n-(i-1) && j<=n+(i-1))
    {
     printf(" * ");
    }
  else
    {
     printf(" ");
    }
```

```
int b[] ={6,2,1,6,3,4,7,8,9,0}
int (*ptr)[10] = &//pointer to an integer array
ptr = &b;
ptr++; → Entire array would be skipped
1040
```

Def Dereferencing point to 1-D array

$*ptr = *(\&b)$

$*ptr = b$

$*ptr = \&b[0]$

[$**(ptr)$ → will give value at $b[0]$]

Q Write a program to convert a decimal no. to binary or convert binary to decimal

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

```
int main()
{
 int flag=0;
 int input;

 do{

printf("If you want to convert decimal to
 Binary, type 1 & If you want to convert
 Binary to decimal, then type 2\n");
    scanf("%d", &input);
    if (input == 1)
      {
       flag=1;
int dec, i, n, a[100];
printf("Enter the decimal no.:\n");
scanf("%d", &dec);
    n=dec;

for(i=0; dec>0; i++)
  {

    dec=dec/2;
    a[i]=dec
    a[i]=dec%2;
    dec=dec/2;
     }

printf("Binary of %d is \n", n);

for(i=i-1; i>=0; i--)
```

```
{
printf("%d", a[i]);
}
}

else if (input==2)
{
  flag = 1;
int bin, n, temp, sum=0;
printf("Enter the binary no:\n");
scanf("%d", &bin);
n=bin;
for (int i=0; n>0; i++)
{
   temp=(n-((n/10)*10))* pow(2,i);

   n= n/10;
   sum= sum + temp);
}
printf("\n Decemal of %d is %d", bin,
                                  sum);
}
else
{
flag=0;
printf("\n Wrong choice. Enter Again");
}
} while (flag==0);
  return 0;
}
```

Output

If you want to convert Decimal to Binary, type 1 & If you want to convert Binary to Decimal=>

1

Enter the decimal no.: 299

Binary of 299 is 100101011

If twin primes are consecutive odd numbers, both of which are prime no.s. Write a program which inputs 2 pos. integers A & B and outputs of all twin primes betn A & B

```
#include <stdio.h>
#include <conio.h>
int main()
{
int a, b;
printf("Enter two positive numbers: ");
scanf("%d %d", &a, &b);
int n[100], k=0, count=0, i, j;
for(i=a; i<=b; i++) /* calling all no. of betn A to B */
{
count=0;
for(j=2; j<i; j++) /* for loop for checking whether no. is prime or not */
{
if(i%j==0)
```

```
    {
        count++;
    }
}
if(count==0)/* storing all prime nos. in desired array */
    {
    n[k]=i;
    k++;
    }
}
for(int i=0; i<=k; i++)
    {
    if(n[i+1]-n[i]==2)
    {
    printf("\n %d %d \n",n[i],n[i+1]);
    }
    }
return 0;
}
```

Output:

```
Enter two positive numbers:
3
99
3  5
5  7
11  13
17  19
29  31
41  43
59  61
71  73
```

Q Write a program to find out ~~whether~~ the sum of proper divisors of a given number "n". The proper divisors of a number "n" are the numbers less than n that divide it; they do not include n itself

```
#include <stdio.h>
int divSum(int n)
{
  int sum=0, i;
for(i=1;i<n;i++)
  {
   if(n%i==0)
    {
      sum+=i;
    }
  }
  return sum;
}

int main()
{
int n;
printf("Enter a number:");
scanf("%d",&n);
printf("sum of ~~proper~~ proper ~~divisors~~ of
                %d is %d", n, divSum(n));

    return 0;
}
```

Output

Enter a number: 12

Sum of proper divisors of 12 is 16.

Q Write a function which takes range as input. Print all the numbers in the range with '*' in front of prime numbers only

```
#include <stdio.h>
void prime(int n1, n2)
{
int i, j, primchk = 1;

for (i=n1; i<=n2; i++)
{
  for (j=2; j<i; j++)
    {
    if (i%j==0){
      primchk = 0;
      break;
      }
  }
  if (prim_chk==1){
    printf("%d*", i);
  }
  else{
    printf("%d", i);
  }
  primchk primchk=1;
  }
}
int main()
  {
  int n1, n2;
  printf("Enter first number: ");
  scanf("%d", &n1);
```

```
printf("Enter last number: ");
scanf("%d", &n2);
prime(n1, n2);
 return 0;
}
```

Output

```
Enter first number = 3
Enter last number = 37

3* 4 5* 6 7* 8 9 10 11* 12 13* 14 15 16 17*
18 19* 20 21 22* 23* 24 25 26 27
28 29* 30 31* 32 33 34 35 36 37*
```

Q3 Write a function which takes as parameters two positive integers and returns TRUE if the numbers are amicable & FALSE otherwise. A pair of numbers is said to be amicable if the sum of divisors of each of the numbers (excluding the no.) itself is equal to the other number

E.g. 220

```
#include <stdio.h>

int amicable(int n1, int n2)
{
int sum1=0, sum2=0, i;
for(i=1; i<n1; i++)
{
 if(n1%i==0)
 {
  sum1 += i;
 }
}
```

```
for (i=2; i<n2; i++)
{
  if (n2%i==0)
  {
    sum2 += i;
  }
}
if (sum1==n2 && sum2==n1)
{
  return 1;
}
else { return 0; }
}

int main()
{
int n1, n2, tf;
printf("Enter first number: ");
scanf("%d", &n2);
printf("Enter second number: ");
scanf("%d", &n2);
if (tf==1){
  printf("%d and %d are amicable
      numbers", n1, n2);
  }
  else {
printf("%d and %d are not amicable
      numbers", n1, n2);
  }
return 0;
}
```

Output: Enter first number: 1184
Enter second number: 1210

1184 and 1210 are amicable numbers

Q4) Write a function to find out whether given numbers are relatively prime or not. 2 numbers is relatively prime if the '1' is the only common factor bet" the 2 nos.

```
#include <stdio.h>
void relPrime(int n1, int n2)
{
int i, rel = 1;
for(i=2; i<n1 || i<n2; i++)
  {
  if(n1%i==0 && n1%i==n2%i)
    {
    printf("%d and %d are not relative
            primes", n1, n2);
      rel=0;
      break;
    }
  }
if(rel==1)
  {
  printf("%d and %d are relative
          primes", n1,n2);
  }
}
```

```
int main()
{
    int n1, n2;
    printf("Enter the first number: ");
    scanf("%d", &n1);
    printf("Enter the second number: ");
    scanf("%d", &n2);
    cal prime(n1, n2);
    return 0;
}
```

Output

Enter first number: 9

— " — second — " — : 8

9 and 8 are relative primes