//1. Problem: Can`t print all the values assigned to variable x.

```
#include <stdio.h>
int main( )
{
        int x ;
        x = 5 ;
        x = 10 ;
        printf ( "\nx = %d", x ) ;
}
```

//Solution: Arrays

//Eg-1 of 1D Array|To find the avg-marks obtained by 30 stds of a class

```
#include<stdio.h>
int main( )
{
        int avg, sum = 0 ;
        int i ;
        int marks[30] ; /* array declaration */
        for ( i = 0 ; i <= 29 ; i++ )
        {
                printf ( "\nEnter marks " ) ;
                scanf ( "%d", &marks[i] ) ; // store data in array */
        }
        for ( i = 0 ; i <= 29 ; i++ )
                sum = sum + marks[i] ; // read data from an array*/

        avg = sum / 30 ;
        printf ( "\nAverage marks = %d", avg ) ;
}
```

// Imp-Array Initialization Examples | In 1-D Array

// Meaning? Address Size

```
int num[6] = { 2, 4, 12, 5, 45, 5 } ; //Memory size for each elements in array
int n[ ] = { 2, 4, 12, 5, 45, 5 } ;
float press[ ] = { 12.3, 34.2 -23.4, -11.3 } ;
```

Note the following points carefully:

(a) Till the array elements are not given any specific values, they are supposed to contain garbage values.

(b) If the array is initialized where it is declared, mentioning the dimension of the array is optional as in the 2nd example above.

//2 Passing Array elements to a function | Call by Value | Ref
//Value: Pass the array of elements
//Ref: Pass the Address of array elements

//* Demonstration of call by value */ (--81)

```
#include<stdio.h>
void display(int);
int main( )
{
        int i ;
        int marks[8] = { 55, 65, 75, 56, 78, 78, 90 } ;
        for ( i = 0 ; i <= 7 ; i++ )
                display ( marks[i] ) ; //value passed
        return 0;
}
void display ( int m )
{
        printf ( "%d ", m ) ;
}
```

//3 /* Demonstration of call by Reference */(--82)

```
#include<stdio.h>
void display(int *); //asterisk
int main( )
{
        int i ;
        int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
        for ( i = 0 ; i <= 6 ; i++ )
                display ( &marks[i] ) ; //add passed
        return 0;
}
void display ( int *n )
{
        printf ( "%d ", *n ) ; //Value at address operator
}
```

//4 /2D Array (Matrix i.e. Grid) (--94)

```
#include <stdio.h>
int main( )
{
        int stud[4][2] ;
        int i;
        for ( i = 0 ; i <= 3 ; i++ )
        {
                printf ( "\n Enter roll no. and marks" ) ;
                scanf ( "%d %d", &stud[i][0], &stud[i][1] ) ; //read and store values
        }
        for ( i = 0 ; i <= 3 ; i++ )
                printf ( "\n%d %d", stud[i][0], stud[i][1] ) ; //print out values
        return 0;
}
```

//4 b/ 2D Array Hardcoded Elements

```
#include <stdio.h>
int main( )
{
        int stud[4][2] = {
                                        { 1234, 56 },
                                        { 1212, 33 },
                                        { 1434, 80 },
                                        { 1312, 78 }
                                };
        int i,j;

        for ( i = 0 ; i <= 3 ; i++ )
                for (j = 0; j < 2; j++)
                        printf ( "\n%d %d", stud[i][j], stud[i][j++] ) ; //print out values
        return 0;
}
```

```
// /5 How to Initialize a 2-Dimensional Array
        int stud[4][2] = {
                                                { 1234, 56 },
                                                { 1212, 33 },
                                                { 1434, 80 },
                                                { 1312, 78 }
                                            } ;
```

or even this would work...

```
        int stud[4][2] = { 1234, 56, 1212, 33, 1434, 80, 1312, 78 } ;
```

```
// Perfectly okay
        int arr[2][3] = { 12, 34, 23, 45, 56, 45 } ;
        int arr[ ][3] = { 12, 34, 23, 45, 56, 45 } ;
```

Not okay

```
        int arr[2][ ] = { 12, 34, 23, 45, 56, 45 } ;
        int arr[ ][ ] = { 12, 34, 23, 45, 56, 45 } ;
```

```
// Example 6 | 2D Array
// /* Demo: 2-D array is an array of arrays */
        #include <stdio.h>
        int main( )
        {
                int s[4][2] = {
                                                { 1234, 56 },
                                                { 1212, 33 },
                                                { 1434, 80 },
                                                { 1312, 78 }
                                            } ;
                int i ;
                for ( i = 0 ; i <= 3 ; i++ )
                        printf( "\nAddress of %d th 1-D array = %u", i, s[i] ) ;
                return 0;
        }
```