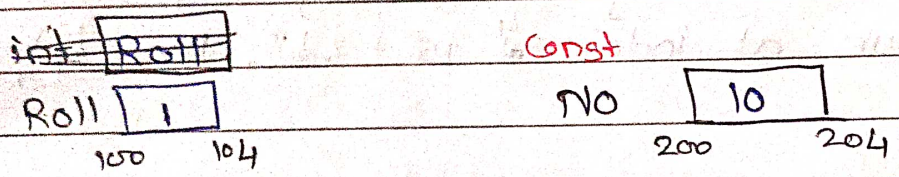
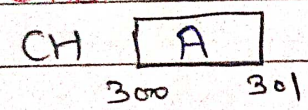
Assignment No-4

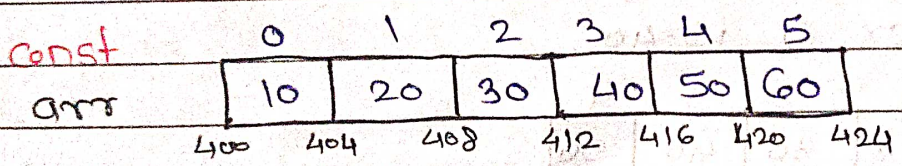
Q1] Read below statement and draw its diagramatic layout.

```
int Roll = 1;
const int NO = 10;
Const CH = 'A';
Const int arr [6] = {10, 20, 30, 40, 50, 60};
```

→ Diagramatic layout







Q2] Explain the concept of one diamensional array with example, diagramatic layout.

→ 1dimensional array is a data structure that holds a fixed number of elements.

- All of the same type, stored in contiguous memory locations.
- Each element in the array can be accessed by its index.

The index of the array start from 0, meaning the first element is at index0, and so on.

- All elements are stored linearly, making it easy to perform operations like access, modify and iterate through the array.

Syntax-

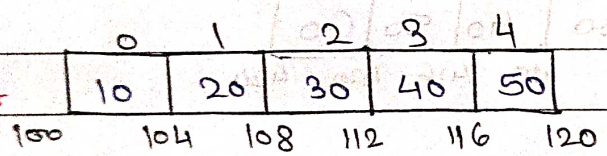
```
type arrayName [size];
```

eg.

```
#include<stdio.h>
int main()
{
  int Arr[5] = {10, 20, 30, 40, 50};
  printf("%d\n", Arr[3]);
  printf("Size of array is %d bytes", sizeof(Arr));
  printf("Value at index 4 is = %d", Arr[4]);
return 0;
}
```

Output-

```
40
Size of array is 20 bytes.
Value at index 4 is=40.
```



Q3] Explain Concept of 2D Array with example & diagramatic layout.

→ A 2D array is an array of arrays, where each element is itself a one dimensional Array.

- It can be visualized as a table or matrix with rows and columns.

Each element in a 2D array is accessed using two indices, one for row and one for column.

Syntax-

```
type array_name [rows] [columns];
```

e.g.

```
#include <stdio.h>
int main()
{
   int arr [3] [3] = {1,2,3,4,5,6,7,8,9};
   arr [0] [0] = 1;
   arr [0] [1] = 2;
   arr [0] [2] = 3;
   arr [1] [0] = 4;
   arr [1] [1] = 5;
   arr [1] [2] = 6;
   arr [2] [0] = 7;
   arr [2] [1] = 8;
   arr [2] [2] = 9;
return 0;
}
```

-Diagramatic layout-

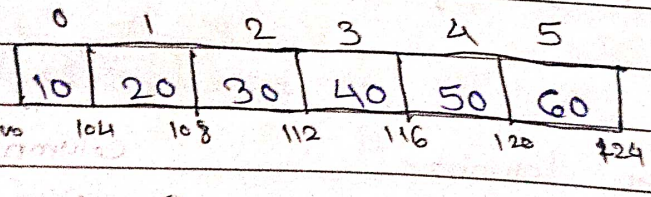
```
      Rows   Columns
arr   [3]    [3]
```

Rows \ Columns	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

	0	1	2
0	[0][0]	[0][1]	[0][2]
1	[1][0]	[1][1]	[1][2]
2	[2][0]	[2][1]	[2][2]

Q4] Write a program to display the content below array, draw the diagramatic layout.
const int arr[6] = {10, 20, 30, 40, 50, 60}

```
#include <stdio.h>
int main()
{
const int arr[6] = {10, 20, 30, 40, 50, 60};
printf ("arr [0] = %d \n", arr[0]);
printf ("arr [1] = %d \n", arr [1]);
printf ("arr [2] = %d \n", arr [2]);
printf ("arr [3] = %d \n", arr [3]);
printf ("arr [4] = %d \n", arr [4]);
printf ("arr [5] = %d \n", arr [5]);
return 0;
}
```



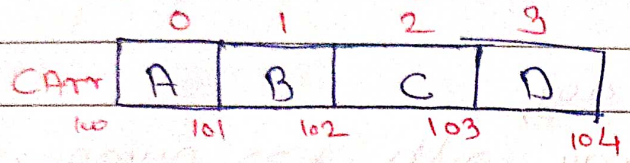
Q5] Write a program to create an array of all primitive data type, draw diagramatic layo

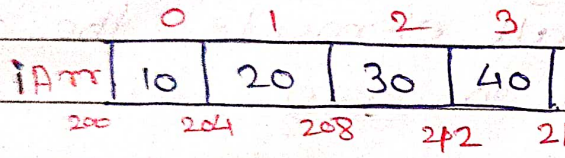
→

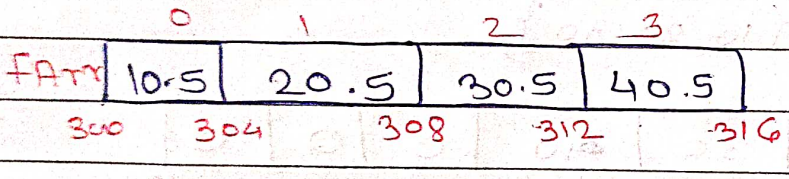
```
#include<stdio.h>
int main()
{
    char cArr[4] = {'A', 'B', 'C', 'D'};
    int iArr[4] = {10, 20, 30, 40, 50};
    float fArr[4] = {10.5, 20.5, 30.5, 40.5};
    double dArr[4] = {10.0, 20.0, 30.0, 40.0};
    printf("size of array is %d bytes", sizeof(cAr
    printf("size of iArr is %d bytes", size of(iArr
```

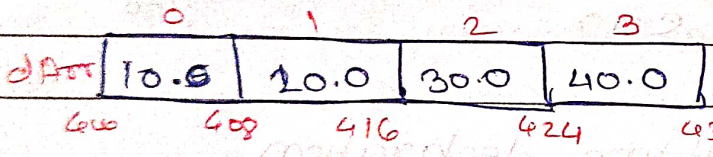
```
    printf("size of fArr is %d bytes", sizeof(fArr));
    printf("size of dArr is %d bytes", sizeof(dArr));
return 0;
}
```

→ Diagramatic layout-









Q6] Explain the different approaches (way) to create the array in C/C++.

→ There are multiple way in which we create and initialized the array.

① With size and Initialization list.

Create the array by specifing the length of array as well as initializing the array. Immediately using initialization of array.

Syntax-

datatype arrayname [length] = {value1, ... value.n};

e.g.

int Arr[5] = {10, 20, 30, 40, 50};

② Array declaration without length.
We can also let the compiler determine the size of the array based on number of elements provided.

Syntax -

datatype arrayname[] = {value1, ... valuen};

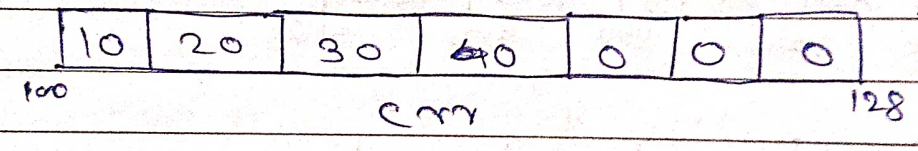
e.g.

int Brr[] = {10, 20, 30, 40};

③ Initializing of array with less numbers
If we don't specify all elements, the remaining elements are initialized to zero.

e.g.

int Crr[7] = {10, 20, 30};



④ Initialization after declaration

e.g. int Drr[3];

Drr[0] = 10;

Drr[1] = 20;

Drr[2] = 30;

Q 7] Draw diagramatic layout and matrix of below array.

int arr[3][3] = {10, 20, 30, 40, 50, 60, 70, 80, 90};

→

arr[3][3] -

	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]

arr [0][0] = 10

arr [0][1] = 20

arr [0][2] = 30

arr [1][0] = 40

arr [1][1] = 50

arr [1][2] = 60

arr [2][0] = 70

arr [2][1] = 80.

arr [2][2] = 90

Q8] Note down the properties of array in C/C++.

→ -Fixed Size-

The size of an array is determined when it is created and cannot be changed.

It can hold a fixed number of elements of the same type.

-Homogenous Elements-

All elements in an array must be of the same data type.

- Contiguous Memory Allocation

The elements of an array are stored in contiguous memory locations, which allows for fast access using an index.

- Indexing-

Array are typically zero indexed, meaning the first element is accessed with index 0.

Elements are accessed using an index, which refers to their position in the array.

- Multidimensional Support-

Array can have more than one diemension.

e.g. (2D, 3D).

Q.9] Explain the concept of length of Array, size of array.

→ Length of Array-

- The length refers to the total numbers of elements the array can hold
- e.g. if an array can store 10 elements, its length is 10.
- ~~In many languages like~~

Size of an Array.

The size of an array often refers to the amount of memory that the array occupies.

- It is calculated by multiplying the number of elements in the array by the size of each element.
- e.g. array of 10 integers in C & each integer take up 4 bytes, the total size of the array is 10 * 4 = 40 bytes.
- To calculate size using sizeof() to get memory used by the entire array
- sizeof(int) to get the size of single element.