Date. No.

Arrays in C:

Array is a collection of homogeneous data items which are stored in contiguous memory location.

→ Need of array:
When we need to use many variables of the same type.

→ Syntax: datatype nameofarray [size];

For array,
i) Array can be of any type.
ii) size has to be specified and it must be positive integer.
iii) Size can be ~~det~~ changed during runtime

⊛ Initialization of Array:

This can be done in two ways:

(a) At compile time:

→ Declaration and initialization is done at same time.

i) int a[5] = {0, 1, 11, 10, 2};
ii) int a[] = {0, 1, 11, 10, 2, 6}; Here, automatically size is calculated.
iii) int a[5] = {0, 1, 2};
Here, remaining two is 0, 0.
iv) int a[5] = {0, 1, 2, 3, 4, 5, 6}; ⇒ error
can't store more numbers than the size initiated with

Date. No.

(b) At run time:

→ It is done by using loop and scanf function.

→ Memory allocation of array is contiguous & memory allocation is one after another.

→ Index of array always starts from 0 and till (size-1)

Eg: int a[5] ⟹ a[0], a[1], a[2], a[3], a[4]

⇓ ⇒

internal pointer (because it stores address) — base address (address of first element).

Eg:
```
#include <stdio.h>
void main()
{ float marks[5], sum=0, avg=0; int i;
 for (i=0; i<5; i++)
 { scanf("%d", &marks[i]); }
 for (i=0; i<5, i++)
 { sum = sum + marks[i]; }
 avg = sum/5;
 printf("Sum = %f"; sum);
 printf("Avg = %f"; avg);
}
```

Output:
61.88
78.14
83.60
44.81
66.66
Sum = 335.030000
Avg = 67.006000

Date. No.

⊛ Passing Array as an Argument to function.

→ It is pass by reference method.

(i): Declaration:

Syntax: returntype functionname (datatype [], datatype);

We can just specify the datatype of array and datatype of size of array while passing 1-d array.

(ii) Calling:

Syntax: functionname (variable name, size);

While calling the function, base address is to be passed along with size of array

(iii) Definition:

Syntax: returntype functionname (datatype array[], datatype size)

While defining function, the array name and size has to be specified.

Example:

```
#include <stdio.h>
int avg(int [], int );
void main( )
{ int average, size;
  int marks [5] = {10, 15, 20, 30, 45};
  size = sizeof(marks) / sizeof (mark[0]);
  average = avg(marks, size);
  printf(" Average = %d", average);
}
```

```
int avg (int marks[], int size);
{ int sum=0, average2;
  for (i=0; i< size ; i++ )
  { sum = sum + marks[i]; }
  average2 = sum/size
  return average2;
}
```

(*) Two-D Array:

The array with row and column is twodarray.

→ Syntax: datatype variablename [rowsize][columnsize]

Initialization:

(a) At compile time:

→ int a[2][3] = {0,0,0,1,1,1}

So, in memory

	0	1	2
0	0	0	0
1	1	1	1

data is filled row by row.

→ int a[2][3] = {{0,0,0},
{1,1,1}}

→ int a[][3] = {{0,0,0}
{1,1,1}
}

No need to initialize row size as the array is implicitly defined.

(b) At run time:

We use nested loop and scanf function.

→ If we provide insufficient values, it remaining spots are filled by 0.

→ If more values are provided, it is declared error.

(4) Passing 2-d array to function:

While passing 2-d array (multidimensional), we only pass the name of multidimensional array

While passing 2-d array, we must specify the numbers of columns but specifying rows is optional.

→ Declaration: returntype functionname (datatype arrayname [ ] [c-s];
→ Calling: functionname (arrayname);
→ Definition: returntype functionname (datatype arrayname [ ] [c-s]

Here, stating row size is not necessary.