Q-2 How to pass an array using call by value in C ? Assuming array is declared locally that is inside main() ?

Solution- We know that, in C when we pass an array to a function , it is passed by reference only means we are actually passing pointer to the first element of the array.

But if we want to pass array by value i.e, we don't want values of original array to change if any changes made in function.

Here I am suggesting **two** approaches-

Approach 1- if our requirement is to not change values of original array.

We can create a temporary array(copy) and pass it to the function.

Code-

```
#include <stdio.h>

// Function to print the elements of an array
void changeArray(int arr[], int size) {
   for (int i = 0; i < size; i++) {
      arr[i]=arr[i]*2;
   }
}
int main() {
   int originalArray[] = {1, 2, 3, 4, 5};
   int size = sizeof(originalArray) / sizeof(originalArray[0]);

   // creating a Copy of the original array
   int copiedArray[size];
   for (int i = 0; i < size; i++) {
      copiedArray[i] = originalArray[i];
   }

   // Pass the copied array to the function
   changeArray(copiedArray, size);

// print both original and copied array and map the difference
  for (int i = 0; i < size; i++) {
      printf("%d ", originalArray [i]);
   }
printf("\n");
for (int i = 0; i < size; i++) {
      printf("%d ", copiedArray [i]);
   }
   return 0;
}
```

Here we can see that value only change in copied array.

Approach 2(more close to problem statement)-

We will create structure containing array, in main function we will create one object to this structure and assign values to it. Now we can pass this object to the function.

Code-

```
#include <stdio.h>
#define SIZE 5

// Define a struct to hold the array
typedef struct {
    int arr[SIZE];
} ArrayStruct;

// Function that takes the struct by value
void modifyArray(ArrayStruct temp) {
    for (int i = 0; i < SIZE; i++) {
        temp.arr[i] *= 2; // Double each element
    }
}

int main() {
    ArrayStruct temp;

    // Initialize the array
    for (int i = 0; i < SIZE; i++) {
        temp.arr[i] = i + 1;
    }

    // Print the original  array
    for (int i = 0; i < SIZE; i++) {
        printf("%d ", temp.arr[i]);
    }
    // Pass the struct by value to the function
    modifyArray(temp);
```

```
// Print the modified array
    for (int i = 0; i < SIZE; i++) {
        printf("%d ", temp.arr[i]);
    }
    printf("\n");

    return 0;
}
```

Here output will be- array before function calling- [1 2 3 4 5] array after function calling- [1 2 3 4 5]