

Part VI: Pointers

The following functions are one of the most common algorithms you can make in C and are a good excuse to practice pointers. By the way most of them, are interview questions

1.- A function that calculates the average of the values in an array

```
unsigned char average( unsigned char *array, unsigned char size );
```

The function should return the average value of the elements in an array, the result must be returned as an integer value, it is not necessary to have precision with decimal values

- **array.**- *a pointer that receives the address where the array with the values to be averaged is located*
- **size.**- *array size*

2.- A function that copies an array to another array

```
void arrayCopy( unsigned char *arrayA, unsigned char *arrayB, unsigned char size );
```

The function should copy the contents of one array and place it in another array in such a way that after calling the function both arrays have the same elements.

- **arrayA.**- *pointer that receives the address of the array with the information to copy*
- **arrayB.**- *pointer that receives the address of the array to which to copy the information*
- **size.**- *number of elements to copy*

3.- Function to compare two arrays

```
unsigned char arrayCompare( unsigned char *arrayA, unsigned char *arrayB, unsigned char size );
```

La funcion aceptara dios arreglos y determinara si ambos tiene la misma informacion, Regresara un valor de 0 si son iguales, y un valor de 1 si son diferentes

- **arrayA.**- *pointer that receives the address of one of the arrays to compare*
- **arrayB.**- *pointer that receives the address of the second array to compare*
- **size.**- *number of elements to compare*

4.- Function to get the largest number in a given array

```
unsigned short largest( unsigned short *array, unsigned char size );
```

The function must return the highest value number in an array.

- **array.**- *a pointer that receives the address where the array with the values is located*
- **size.**- *array size*

5.- Function to sort an array of a given size

```
void sortArray( unsigned short *array, unsigned char size );
```

the function must order the elements in an array from smallest to largest

- **array.**- *a pointer that receives the address where the array with the values to be sorted is located*
- **size.**- *array size*