#6 Suppose a **homogeneous** **unbounded** array of **any data type** is represented in **C** using the following typedef:

typedef struct

{

short shElementSize; // size of a single element in the array

short iAllocatedElements; // number of allocated elements

int iHighestPopulatedElement; // highest subscript of a populated element

//ex(length of array IS ZERO Terminated)

void \*pArray; // pointer to memory containing a contiguous array

} ARRAYSTRUCT;

Show code for the C function, **int reAllocate**(ARRAYSTRUCT \*pStruct), which should

* allocate memory (use malloc or calloc) for the new array, doubling its current size
* copy the old data into the new array
* set the attributes in the structure
* functionally returns TRUE if the reallocate was successful; otherwise it functionally returns FALSE
* what else is needed?

Int reAllocate(ARRAYSTRUCT \*pStruct) {

Void \*pOld = pStruct ->pArray;

//pArray points to doubled size array

pStruct->pArray = malloc(pStruct->shElementSize\*pStruct->iAllocatedElements \* 2);

if (pStruct->pArray == NULL)

return FALSE

//pStruct allocated elements is doubled

pStruct->iAllocatedElements = pStruct->iAllocatedElements \* 2;

//set all values in array to null

memset(pStruct->pArray, ‘\0’, pStruct->iAllocatedElements \* pStruct->shElementSize);

//copy old data into new array

memcpy(pStruct->pArray, pOld, (pStruct->shElementSize)\*(pStruct->iHighestPopulatedElement+1));

free(pOld);

return TRUE;

}