Instrucciones: lee detenidamente cada pregunta y responde de manera correcta; aplicando tus conocimientos del capítulo 6 del libro de Deitel; correspondientes a las secciones 6.6 al 6.8.

✓ ¿Cuál de los algoritmos de ordenación tiene una complejidad igual a **(n*log2*n)**? 5/5

- (●) QuickSort. ✓
- ( ) Burbuja.
- ( ) Inserción.
- ( ) N.d.a
- ( ) Selección.

✓ Si un algoritmo de ordenación tiene eficiencia logarítmica entonces este algoritmo es _____ que un algoritmo con eficiencia lineal. 5/5

- ( ) Peor.
- ( ) Igual.
- (●) Mejor. ✓

✗ El algoritmo de ordenación que aplica la técnica **divide y vencerás** corresponde a: 0/5

- (●) N.d.a. ✗
- ( ) QuickSort.
- ( ) Selección.
- ( ) Inserción
- ( ) Shell.

Respuesta correcta

- (●) QuickSort.

✓ Para que el algoritmo de ordenación **QuickSort** tenga un alto desempeño depende: 5/5

- (●) De la elección del Pivote. ✓
- ( ) Del tamaño de la lista.
- ( ) De la función del algoritmo sea recursivo.
- ( ) Que la lista se encuentre ordenada.
- ( ) N.d.a.

✓ ¿En qué número de línea se encuentra el error? Para que el siguiente programa cumpla correctamente con la búsqueda lineal en un arreglo. 5/5

```
1  #include <stdio.h>
2  #define TAMANIO 100
3
4  int busquedaLineal( const int arreglo[], int llave, int tamanio );
5
6  int main()
7  {
8      int a[ TAMANIO ];
9      int x;
10     int llaveBusqueda;
11     int elemento;
12
13     for ( x = 0; x < TAMANIO; x++ ) {
14         a[ x ] = 2 * x;
15     }
16
17     printf( "Introduzca la llave de busqueda entera:\n" );
18
19     scanf( "%d", &llaveBusqueda );
20
21     elemento = busquedaLineal( a, llaveBusqueda, TAMANIO );
22
23     if ( elemento != -1 ) {
24         printf( "Encontre el valor en el elemento %d\n", elemento );
25     }
26     else {
27         printf( "Valor no encontrado\n" );
28     }
29
30     return 0;
31 }
32
33 int busquedaLineal( const int arreglo[], int llave, int tamanio ) {
34     int n;
35
36     for ( n = 0; n > tamanio; ++n ) {
37         if ( arreglo[ n ] == llave ) {
38             return n;
39         }
40     }
41
42     return -1;
43 }
```

36 ✓

✓ Utilizando el algoritmo de **BÚSQUEDA BINARIA.** ¿Cuántas pasadas se necesita para encontrar el valor **88**? Para la lista de abajo: 5/5

[8, 13, 17, 26, 44, 56, 88, 97]

- ○ 4
- ◉ 3 ✓
- ○ 1
- ○ 2
- ○ 0

✓ ¿Qué algoritmo de ordenación se está usando? 5/5

Si inicialmente, se tiene la siguiente lista:

[47, 3, 21, 32, 56, 92]

Después de dos pasadas de un algoritmo de ordenación, la lista se ha quedado dispuesto así:

[3, 21, 47, 32, 56, 92]

- ○ QuickSort.
- ◉ Inserción. ✓
- ○ Selección.
- ○ Burbuja.
- ○ N.d.a.

✓ Cuando la variable ***bajo*** es mayor que la variable ***alto*** en una **BÚSQUEDA BINARIA** significa que ya no existe vector a buscar la clave. 5/5

```
1   int busquedaBinaria (int lista[], int n, int clave) {
2
3       int central, bajo = 0, alto = n - 1, valorCentral;
4
5       while (bajo <= alto) {
6           ...
7       }
8
9       ...
10  }
```

- ○ Falso.
- ◉ Verdadero. ✓

✓ Este algoritmo en la primera pasada busca el menor elemento desde ***v[0]*** a ***v[n - 1]*** y lo deposita en la posición ***v[0]***, luego se busca el menor elemento desde ***v[1]*** a ***v[n - 1]*** y lo deposita en la posición ***v[1]*** y, así sucesivamente para las siguientes pasadas del algoritmo. 5/5

- ○ Inserción.
- ○ QuickSort.
- ○ Shell.
- ○ Burbuja.
- ◉ Selección. ✓
- ○ N.d.a.

✗ Si se tiene una lista de 5.000 elementos el número de elementos examinados en el peor de los casos en una **búsqueda binaria** seria: 0/5

- ○ 14
- ○ 11
- ○ 41
- ○ 18
- ◉ 81 ✗

Respuesta correcta

- ◉ 14

✓ La codificación de la **BÚSQUEDA BINARIA** sólo se puede realizar de forma ITERATIVA. 5/5

- ○ Verdadero.
- ◉ Falso. ✓

✓ Si los datos están almacenados en un archivo, el proceso de ordenación se llama ordenación: _____. 5/5

externa ✓

✓ En el método de hundimiento para ordenar arreglos, el ciclo interno cumple la función para controlar el número de _____ por pasada. 5/5

comparaciones ✓

✓ ¿Cuál es el algoritmo de ordenación que mejora el método de **Inserción Directa**? 5/5

- ○ Selección.
- ◉ Shell. ✓
- ○ Insercción.
- ○ QuickSort.
- ○ N.d.a.

✓ Para que se aplique correctamente la **BÚSQUEDA BINARIA** la lista NO tiene que estar ordenada ya sea de forma ascendente o descendente. 5/5

- ○ Verdadero.
- ◉ Falso. ✓

✓ El algoritmo que se basa en las lecturas sucesivas de la lista a ordenar, comparando el elemento inferior de la lista con los restantes y efectuando _____ de posiciones cuando el orden resultante de la comparación no sea el correcto. 5/5

1. recorrido
2. inserción
3. intercambio ✓
4. mezcla
5. selección

✓ En terminología de ordenación el elemento por el cual está ordenado un conjunto de datos (o se esta buscando) se denomina: _____. 5/5

clave ✓

✓ La eficiencia en una **BÚSQUEDA SECUENCIAL** es: 5/5

- ○ log2n (logaritmo de base 2 por n)
- ○ N.d.a.
- ○ 1
- ◉ n ✓
- ○ n2 (n al cuadrado)

✗ ¿Este programa ordena los valores de un arreglo en orden? 0/5

```c
#include <stdio.h>
#define TAMANIO 10

int main()
{
    int a[ TAMANIO ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };

    int pasadas;
    int i;
    int almacena;

    printf( "Elementos de datos en el orden original\n" );

    for ( i = 0; i < TAMANIO; i++ ) {
        printf( "%4d", a[ i ] );
    }

    for ( pasadas = 1; pasadas < TAMANIO; pasadas++ ) {

        for ( i = 0; i < TAMANIO - 1; i++ ) {

            if ( a[ i ] < a[ i + 1 ] ) {
                almacena = a[ i ];
                a[ i ] = a[ i + 1 ];
                a[ i + 1 ] = almacena;
            }
        }
    }

    printf( "\nElementos de datos en orden ascendente\n" );

    for ( i = 0; i < TAMANIO; i++ ) {
        printf( "%4d", a[ i ] );
    }

    printf( "\n" );

    return 0;
}
```

- ○ Ascendente.
- ○ Descendente.
- ○ Tiene error de sintaxis.
- ◉ No ordena la lista. ✗

Respuesta correcta

◉ Descendente.

✓ La búsqueda _____ compara cada elemento de un arreglo con la *clave de búsqueda*. 5/5

lineal ✓