

Prueba 1

2S - 2015

NOMBRE:

NRO.MATRICULA :

☐ Estructura de Datos ☐ Complejidad Computacional

Programación Orientada a Objetos

1. Escriba los métodos para mover la pelota en el plano X-Y:

```
public class Pelota {  
    int posX;  
    int posY;  
  
    public Pelota() {  
        posX = 0;  
        posY = 0;  
    }  
  
}
```

}

2. ¿Qué valor de z imprime este código? [3 ptos]

```
boolean go = true;  
int x=5;  
int y=1;  
int z=4;  
while(go) {  
    z += x++ - y;  
    if(y >= 5) go = false;  
    else y++;  
}  
System.out.println("z:"+z);
```

z = _____

Arreglos

3. Insertar un elemento en un arreglo inversamente ordenado [1 pto]
- a. Toma el mismo tiempo independiente del número de elementos actuales.
 - b. Requiere N^2 comparaciones
 - c. Requiere mover $\log(N)$ elementos.
 - d. Toma un tiempo proporcional a tamaño del arreglo
 - e. Ninguna de las anteriores
4. En un arreglo desordenado, es generalmente más rápido encontrar un elemento que no esta en el arreglo, a encontrar un elemento que se encuentra en el. JUSTIFIQUE. [2 ptos]

a. verdadero

b. falso

Porque, _____

5. Los arreglos ordenados, comparados con los arreglos desordenados [1 pto]
- a. Ocupan mas espacio en memoria
 - b. Son más rápidos de crear.
 - c. Pueden crecer dinámicamente.
 - d. Son más rápidos en la inserción.
 - e. Ninguna de las anteriores
6. Ordene los siguientes tiempos de ejecución del más eficiente al menos eficiente: $O(N^3)$, $O(N^2)$, $O(N \log N)$, $O(\log N)$, $O(N)$. [2 ptos]
- a. $O(\quad)$
 - b. $O(\quad)$
 - c. $O(\quad)$
 - d. $O(\quad)$

7. De que orden es el siguiente algoritmo [2 ptos]

<pre>a=5 b=6 c=10 for i in range(n): for j in range(n): x = i * i y = j * j z = i * j for k in range(n): w = a*k + 45 v = b*b d = 33</pre>		$O(\quad)$
--	--	--------------

8. La notación ***O*** indica [1 pto]

- a. El tiempo de ejecución de un algoritmo para el tamaño de una estructura de datos determinada.
- b. Cómo se relaciona la velocidad de un algoritmo al número de ítems.
- c. El tiempo en segundos que tarda un algoritmo en procesar un numero determinado de ítems.
- d. Como el tamaño de la estructura de datos se relaciona con el número de ítems.
- e. Ninguna de las anteriores

9. Dada la siguiente clase. Implemente el método insertar. No olvide incluir los comentarios respectivos al código según corresponda. [6 ptos]

```
class arregloOrdenado {
    private long[] a;
    private int nElems;

    public arregloOrdenado (int max) {
        a = new long[max];
        nElems = 0;
    }
}
```

```
public void insertar (long valor)
{
}
}
```

Ordenamiento Simple

10. Los algoritmos de ordenamiento de computadores son más limitados que el ordenamiento hecho por humanos porque: [1 pto]
- Los humanos saben lo que hay que ordenar, mientras que las computadoras necesitan una especificación de lo que hay que ordenar.
 - Los computadores sólo pueden manejar una cantidad limitada de datos.
 - Los computadores sólo pueden comparar dos cosas a la vez.
 - Ninguna de las anteriores

11. ¿En que caso el ordenamiento por Inserción podría ser menos eficiente que el la Burbuja? [1 pto]

12. ¿Por que el ordenamiento por Selección es mejor que el ordenamiento por Inserción? [1 pto]

13. Describa los pasos necesarios para ordenar el arreglo $A = [3, 2, 1]$ usando el Algoritmo de *Inserción*. Explique en cada paso cual fue la operación realizada (copiar <valor>, mover <valor>) [6 ptos]

3	2	1
---	---	---

--	--	--

--

--	--	--

--

--	--	--

--

--	--	--

--

--	--	--

--

--	--	--

--

--	--	--

--

14. Identifique el nombre del algoritmo [4 ptos]

```
public void A() {  
    int in, out;  
    for(out=1; out<nElems; out++) {  
        long temp = a[out];  
        in = out;  
        while(in>0 && a[in-1] >= temp) {  
            a[in] = a[in-1];  
            --in;  
        }  
        a[in] = temp;  
    }  
}
```

```
public void B(long value) {  
    int j;  
    for(j=0; j<nElems; j++)  
        if(a[j] > value)  
            break;  
    for(int k=nElems; k>j; k--)  
        a[k] = a[k-1];  
    a[j] = value;  
    nElems++;  
}
```

```
public int D(int L, int R, long p) {  
    int LP = L - 1;  
    int RP = R + 1;  
    while(true) {  
        while(LP < R && theArray[++LP] < p);  
        while(RP > L && theArray[--RP] > p);  
        if(LP >= RP) break;  
        else swap(LP, RP);  
    }  
    return LP;  
}
```

```
public int C(long value) {  
    int LB = 0;  
    int UB = nElems-1;  
    int curIn;  
    while(true) {  
        curIn = (LB + UB) / 2;  
        if(a[curIn] == value) return curIn;  
        else if(LB > UB) return nElems;  
        else {  
            if(a[curIn] < value) LB = curIn + 1;  
            else UB = curIn - 1;  
        }  
    }  
}
```