Departamento de Ingeniería Telemática
Grado en Ingeniería de Sistemas de Comunicaciones
Programación de Sistemas
Leganés, 4 de mayo de 2018

Examen Parcial II (teoría)
Modelo A

Duración de la prueba: hasta 20 minutos
Puntuación: 3 puntos sobre 10 del examen

Nombre:	
Grupo (61/62):	

NIA:

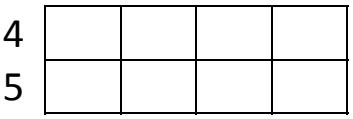
Firma

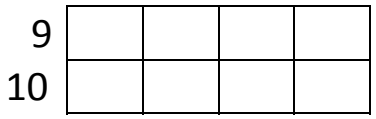
En cada pregunta, sólo una opción es correcta. Cada respuesta correcta suma 0.3 puntos. Cada respuesta incorrecta resta 0.1 puntos. Las preguntas sin contestar no suman ni restan puntos.

- Marca la respuesta a cada pregunta con una equis ("X") en la tabla de abajo.
- Si marcas más de una opción o ninguna opción, la pregunta se considera no contestada.

	A	B	C	D
1				
2				
3				

	A	B	C	D
6				
7				
8				





1. ¿Qué calcula el siguiente método (dado p>=0)?

```
public static int sorpresa(int p, int q){
      if (p == 0){
            return q;
      } else{
            return sorpresa(p-1,q-1);
      }
}
```

   A. q
   B. p + q
   C. p - q
   D. q – p (***)

2. Dado el siguiente método, ¿qué valor regresa la llamada:sorpresa(5)?

```
public static int sorpresa(int n){
      if (n <= 1){
      return 1;
      } else {
      return n * sorpresa(n-2);
      }
}
```

   A. 1
   B. 10
   C. 25
   D. 15 (***)

3. A continuación te mostramos parte de las clases “Node<E>” y “MyFirstLinkedList<E>”:

```
public class Node<E>{

      private E info;

      private Node<E> next;

      public Node<E> getNext() { ... }

      public void setNext( Node<E> next ) { ... }

}

public class MyFirstLinkedList<E> {

            protected Node<E> first;

            public void deleteFirstNode(){  // POR HACER }

}
```

Si asumimos que la lista *no* está vacía y que no tiene nodos *dummy*, el código para eliminar el primer nodo de la lista (deleteFirstNode()) es:

   A. first.setNext(first.getNext().getNext());
   B. first = first.getNext(); (***)

C. first.setNext(first.getNext());
D. first = last.getNext();

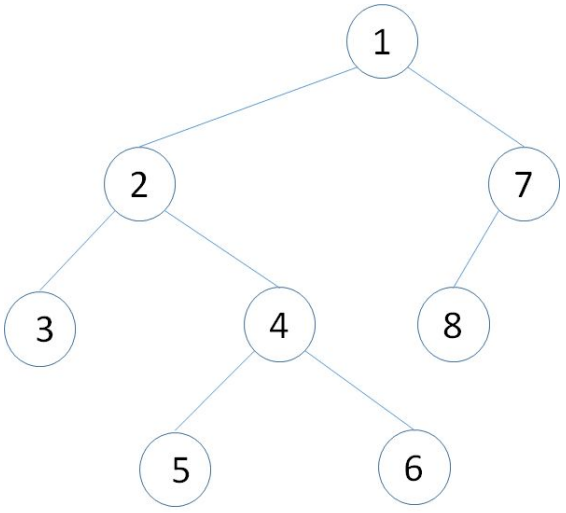
4. Dada una pila inicialmente vacía, indica qué devolvería la llamada al método top() de la pila al terminar de ejecutar la siguiente secuencia de operaciones:

   push(5); top(); pop(); push(2); push(3); pop();

   A. 2 (***)
   B. 3
   C. 5
   D. Ninguna de las otras opciones es correcta.

5. Se realizan las siguientes acciones sobre una cola de Strings vacía como la vista en clase: enqueue("Tánger"); enqueue("Madrid"); enqueue("Albacete"); dequeue(); dequeue();.
   Indica qué se obtiene si a continuación se utiliza el método front();
   A. Albacete (***)
   B. Tánger
   C. Madrid
   D. null o una excepción

6. ¿Cuál sería el resultado de recorrer en postorden el siguiente árbol?



   A. 3 5 6 4 2 8 7 1 (***)
   B. 1 2 3 4 5 6 7 8
   C. 3 2 5 4 6 1 8 7
   D. 1 2 7 3 4 8 5 6

7. Un programador desea programar el método "public String toStringInOrder()" para construir un String del recorrido (visita) de un árbol binario en in-orden. Ya tiene escrita esta parte del método:

```
public String toStringInOrder() {
    if ( root == null ) {  return " "; }
    else {  return ( /* Por Hacer */  ) }
}
```

Nuestro programador sabe que si usa adecuadamente parte de los siguientes fragmentos de código, logrará su propósito:

X: root.getLeft().toString()

Y: root.getRight().toString()

Z: root.getInfo().toString()

W: + " " +

A. X W Y W Z
B. Z W X W Y
C. X W Z W Y (***)
D. Y W X W Z

8. El número de aristas que van desde la raíz de un árbol a la hoja que está más alejada de la raíz del árbol se denomina el/la _____ del árbol.
   A. Tamaño
   B. Altura (***)
   C. Longitud
   D. Ninguna de las otras opciones es correcta

9. El siguiente método pretende sumar los elementos de una lista enlazada. Indica la instrucción que falta en el espacio marcado por puntos suspensivos para completar el método.

Nota:

- La lista contiene números enteros.
- El atributo Node<E> first de la lista referencia a su primer nodo

```
public int sumaLista (){
        int suma = 0;
        for ( ... ){
        suma = suma + actual.getInfo();
        }
        return suma;
}
```

A. Node<E> actual = first; actual.getNext() != null; actual = actual.getNext()
B. Node<E> actual = first; actual != null; actual = actual.getNext() (***)
C. Node<E> actual = first; actual != null; actual = actual.getInfo()
D. Node<E> actual = first; actual != null; actual ++

10. El siguiente método pretende implementar el algoritmo de ordenación burbuja (BubbleSort). Indica la instrucción que falta en el espacio marcado por puntos suspensivos para completar el método de la manera *más eficiente* posible.

```
public static void bubbleSort(int[] a){
        for ( int i=0; i<a.length-1; i++ ) {
         for (  ...  ) {
                  if (  a[j] > a[j+1] ) {
                           swap(a, j, j+1);
                  }
         }
        }
}
public static void swap( int[] a, int i, int j ){
        int aux = a[i];
         a[i] = a[j];
        a[j] = aux;
}
```

A. int j=0; j<a.length; j++
B. int j=0; j<a.length-i; j++
C. int j=0; j<a.length-1; j++
D. int j=0; j<a.length-1-i; j++ (***)