



Programación de Sistemas
Grado en Ingeniería Telemática

Leganés, 10 de mayo de 2019
Duración de la prueba: 20 min

Examen parcial 2 (teoría)
Puntuación: 3 puntos sobre 10 del examen

Sólo una opción es correcta en cada pregunta. Cada respuesta correcta suma 0,3 puntos. Cada respuesta incorrecta resta 0,1 puntos. Las preguntas no contestadas no suman ni restan puntos.

Marca:  Anula:  No uses:   

- 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.
- Rellena **tus datos personales** antes de comenzar a realizar el examen.

Nombre:

Grupo:

Firma:

NIA:

--	--	--	--	--	--	--	--	--	--

	A	B	C	D		A	B	C	D
1					6				
2					7				
3					8				
4					9				
5					10				



- 1.- Dado el siguiente código. Al ejecutar el programa, ¿qué se muestra por pantalla? Nota: % en java es la operación módulo (resto de dividir un número entre otro).

```
public class Recursividad{
    public String m1(int i){
        if (i<0) return "";
        else if ( (i%2) != 0) return "" + m1(i-1);
        else return i + " " + m1(i-1);
    }
    public static void main(String args []){
        Recursividad r = new Recursividad();
        String cadena = r.m1(10);
        System.out.println(cadena);
    } // main
} // Recursividad
```

- (a) *** 10 8 6 4 2 0
- (b) 9 7 5 3 1
- (c) 10 9 8 7 6 5 4 3 2 1 0
- (d) 9 8 7 6 5 4 3 2 1

- 2.- Dada una lista enlazada simple de objetos de la clase *Integer*, no vacía y con más de un elemento. Si se llama al método *m1()* ¿a qué nodo apunta *current* después de salir del bucle *while*?

```
public void m1() {
    Node<E> current = this.first;

    while(current != null) {
        current = current.getNext();
    }
}
```

- (a) *** A ninguno. A null.
- (b) Al último nodo de la lista.
- (c) Al primer nodo de la lista.
- (d) La ejecución da un error *javaLang.NullPointerException*

- 3.- Dada una *LinkedStack<E>* implementada según la interfaz indicada en (A). Sobre una pila vacía de objetos *Integer* se hacen las operaciones indicadas en (B). ¿Qué valor tiene *i* cuando finaliza la ejecución de la última sentencia?

(A)

```
interface Stack<E> {
    boolean isEmpty();
    int size();
    void push (E info);
    E pop();
    E top();
}
```

(B)

```
pila.push(1);
i = pila.top();
pila.push(2);
i = pila.pop();
pila.push(3);
i = pila.pop();
i = pila.pop();
pila.push(4);
i = pila.pop();
i = pila.pop();
i = pila.pop();
pila.push(5);
i = pila.pop();
```

- (a) *** 5
- (b) 6
- (c) 7
- (d) 4

4.- Dada una `LinkedList<E>` y la implementación de su método *dequeue* de la siguiente manera. ¿Cuál de las siguientes afirmaciones es la correcta?

```
public E dequeue(){
    E info;
    if (!isEmpty()) {
        info = top.getInfo();
        top = top.getNext();
        size--;
    } else {
        info = null;
    }
    return info;
}
```

- (a) *** En determinadas ocasiones se va a quedar la cola inconsistente por no actualizar correctamente *tail*
- (b) En determinadas ocasiones se producirá una excepción *java.lang.NullPointerException*

- (c) El método es correcto devolviendo la información y borrándola.
- (d) *dequeue()* no debe borrar el elemento.

5.- En una estructura de datos *deque*, ¿cuál de las siguientes afirmaciones es la correcta?

- (a) *** Es más eficiente con una implementación mediante lista doblemente enlazada.
- (b) Es más eficiente con una implementación mediante lista dinámica simplemente enlazada.
- (c) Con sus métodos *insertLast()* y *removeFirst()* estamos en condiciones de utilizar la *deque* como una *stack*
- (d) Con sus métodos *insertFirst()* y *removeLast()* estamos en condiciones de utilizar la *deque* como una *stack*

6.- Dado el siguiente árbol binario de objetos *Integer*, representado por el siguiente array, en un recorrido preorden ¿cómo se visitarían los nodos?:

{1, null, 2, null, null, null, 3, null, null, null, null, null, null, 6, 4}

- (a) *** 1, 2, 3, 6, 4
- (b) 6, 4, 3, 2, 1
- (c) 1, 2, 6, 3, 4
- (d) 4, 6, 3, 2, 1

7.- Dado el siguiente árbol binario de objetos *Integer*, representado por el siguiente array, ¿es un montículo?

{1, 2, 4, 3, 5, 6, 9, 8, 7, null, null, null, null, null, null}

- (a) *** Sí; es un montículo min-heap.
- (b) No; no es un montículo puesto que no cumple ni min-heap ni max-heap.
- (c) No; no es un montículo porque no es completo.
- (d) Sí; es un montículo max-heap.

8.- Respecto a la eficiencia de los algoritmos de búsqueda estudiados en clase (lineal y binaria), teniendo en cuenta que N es el número de elementos, ¿cuál de las siguientes afirmaciones es la correcta?:

- (a) *** Para valores pequeños de N no merece la pena la búsqueda binaria.
- (b) Para valores muy grandes de N la búsqueda lineal es más eficiente que la binaria.
- (c) La búsqueda lineal tiene una eficiencia de $O(\log N)$
- (d) La búsqueda binaria tiene una eficiencia de $O(N)$

- 9.- ¿Cuántos intercambios hace el algoritmo de ordenación Bubble Sort para ordenar de menor a mayor este array de enteros: {3, 2, 1}?

```
public class BubbleSort {

    public static void bubbleSort (int[] a) {
        int swaps = 0;
        for (int i=0; i<a.length-1; i++) {
            for (int j=0; j<a.length-1-i; j++) {
                if (a[j]>a[j+1]){
                    swap(a, j, j+1);
                    swaps++;
                }
            }
        }
        System.out.println("Number of swaps: " + swaps);
    } // bubbleSort

    public static void swap (int[] a, int i, int j) {
        int aux=a[i];
        a[i]=a[j];
        a[j]=aux;
    }

    public static void main(String[] args){
        int[] array = {3, 2, 1};
        bubbleSort(array);
    }
}
```

- (a) *** 3
- (b) 4
- (c) 2
- (d) 1
- 10.- De las estructuras de datos estudiadas en clase cuál escogerías para permitir un el acceso más rápido a los ordenadores de la Universidad con usuario(NIA)/contraseña tanto del personal como el alumnado.
- (a) *** Árbol binario de búsqueda.
- (b) Stack.
- (c) Queue.
- (d) Linked List.