UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional

EI1008/MT1008 Programación II

Examen final. Primera Convocatoria (3 de junio de 2021)

Apellidos: .....

Nombre: ..... **DNI:** .....

Escribe tu nombre completo en todas las hojas y **entrega por separado**:

- Las preguntas 1 y 2 (responde en el propio enunciado).
- La pregunta 3.
- La pregunta 4.

Pregunta 1 (0,5 puntos)

a) Expresa, empleando la notación $\mathcal{O}$, el tiempo de ejecución en el peor de los casos del siguiente fragmento de código cuando la longitud del vector es n:

```
int i = 0;
int positivos = 0;
int negativos = 0;
while (i < vector.length) {
    while (i < vector.length && vector[i] >= 0) {
        positivos++;
        i++;
    }
    while (i < vector.length && vector[i] < 0) {
        negativos++;
        i++;
    }
}
```

b) Expresa, empleando la notación $\mathcal{O}$, el tiempo de ejecución en el peor de los casos del siguiente método cuando la matriz tiene n filas y n columnas:

```
public static boolean triangular(int[][] matriz) {
    int n = matriz.length;
    for (int i = 0; i < n; i++)
        for (int j = i + 1; j < n; j++)
            if (matriz[i][j] != 0)
                return false;
    return true;
}
```

Pregunta 2 (0,5 puntos)

Dado el método público `condición` definido a continuación:

```
public static boolean condición(String s) {
    return condición(s, 0);
}

private static boolean condición(String s, int i) {
    if (i >= s.length() - 1)
        return true;
    else if (s.charAt(i) > s.charAt(i + 1))
        return false;
    else
        return condición(s, i + 1);
}
```

indica qué muestra la ejecución del siguiente fragmento de código:

```
String[] palabras = {"afijo", "sufijo", "itv", "cara", "cruz", "himno"};
for (String s : palabras)
    System.out.println(s + " --> " + condición(s));
```

Pregunta 3 (4,5 puntos)

Disponemos de un dispositivo que permite medir el tiempo empleado por un nadador en completar un largo de piscina en sus entrenamientos. Para cada sesión de entrenamiento, el dispositivo genera un fichero de texto en el que en cada línea encontramos el tiempo en segundos empleado en completar cada largo. A continuación tienes un ejemplo de fichero (su contenido se muestra en dos columnas para reducir espacio en el enunciado):

```
8  | 50
30 | 28
34 | 39
31 | 26
32 | 9
```

Estamos desarrollando un programa que permitirá almacenar los datos del entrenamiento en una lista de nodos con enlace simple y referencias al primero y al último de los nodos. Aquí tienes parte de su implementación:

```
public class EntrenamientoPiscina {
    private static class Nodo {
        int tiempo;  // Tiempo empleado en un largo
        Nodo sig;    // Referencia al siguiente nodo

        Nodo(int elTiempo, Nodo elSiguiente) {
            this.tiempo = elTiempo;
            this.sig = elSiguiente;
        }
    }

    private Nodo primero;    // Referencia al primer nodo
    private Nodo último;     // Referencia al último nodo
}
```

Añade a la clase `EntrenamientoPiscina` los siguientes constructores y métodos públicos:

a) `void insertar(int segundos)` **(0,75 puntos)**

Añade al final de la lista un nuevo nodo que contenga la cantidad de segundos indicada. Su coste temporal debe ser $\mathcal{O}(1)$.

b) `EntrenamientoPiscina(String nombreFichero) throws FileNotFoundException` **(1 punto)**

Construye la lista de nodos a partir de los datos leídos desde un fichero (como el del ejemplo) cuyo nombre se recibe como argumento. Cada tiempo del fichero se guardará en un nodo de la lista. Este constructor debe llamar al método `insertar` del apartado anterior.

c) `double ritmoMedio()` **(0,75 puntos)**

Devuelve la media de los tiempos empleados en completar cada largo o cero si la lista está vacía. Su coste temporal debe ser $\mathcal{O}(n)$, donde n es la cantidad de elementos de la lista.

d) `void eliminarTiemposFalsos()` **(2 puntos)**

Elimina de la lista aquellos nodos cuyos tiempos son falsas medidas debidas a fallos en el dispositivo de lectura. Se considera que un tiempo es falso cuando la diferencia en valor absoluto[1] entre ese tiempo y el tiempo medio (calculado en el método `ritmoMedio`) es mayor que la mitad del tiempo medio. En el ejemplo, los tiempos de 8, 50 y 9 segundos cumplen esa condición y, por tanto, se consideran tiempos falsos. El coste temporal del método debe ser $\mathcal{O}(n)$, donde n es la cantidad de elementos de la lista.

[1] Recuerda que el método estático `abs` de la clase `Math` permite calcular el valor absoluto de un número.

Pregunta 4 (4,5 puntos)

Queremos realizar una aplicación para que las universidades gestionen las inscripciones a los distintos cursos de verano que organizan. Para ello, ya disponemos de una clase, `Inscripción`, que proporciona los siguientes métodos públicos de consulta:

- `String getCódigoCurso()`: devuelve el código del curso.
- `String getDNI()`: devuelve el DNI de la persona inscrita.

Necesitamos definir una nueva clase, `InscripcionesCursosVerano`, que gestione las inscripciones de una universidad. Considera que ya tenemos la siguiente definición parcial:

```
public class InscripcionesCursosVerano {

    private static final int TAMAÑO_INICIAL = 10;

    private String universidad;    // Nombre de la universidad
    private int año;               // Año de celebración de los cursos de verano
    private Inscripción[] vector;  // Datos de las inscripciones a los cursos de verano
                                   // (no guardan ningún orden concreto)
    private int ocupados;          // Número de inscripciones (componentes ocupadas al principio
                                   // del vector; el resto siempre será null)

    public InscripcionesCursosVerano(String universidad, int año) {
        this.universidad = universidad;
        this.año = año;
        this.vector = new Inscripción[TAMAÑO_INICIAL];
        this.ocupados = 0;
    }
}
```

Añade a la clase `InscripcionesCursosVerano` los siguientes métodos públicos (si lo necesitas, puedes definir otros métodos auxiliares):

a) `void añadirInscripción(Inscripción nueva)` **(1 punto)**

Recibe una nueva inscripción y la añade al vector de inscripciones. Siempre que sea necesario aumentar la capacidad del vector, esta se duplicará.

b) `double porcentajeRepetición(String[] dnis)` **(1,75 puntos)**

Recibe un vector, ordenado lexicográficamente de menor a mayor, con los DNI de todas las personas que se matricularon en algún curso de verano de la universidad en años anteriores. Como resultado, devuelve el porcentaje de inscripciones nuevas en ese año, es decir, inscripciones de personas que no habían realizado anteriormente cursos de verano en esa universidad.

Teniendo en cuenta que el vector `dnis` está ordenado, se exige que el coste temporal de este método sea $\mathcal{O}(n \cdot log\ m)$, donde n es el número de inscripciones (valor del atributo `ocupados`) y m es la cantidad de elementos del vector `dnis`.

c) `String[] cancelar(String códigoCurso)` **(1,75 puntos)**

Elimina del vector de inscripciones todas las que se corresponden con el código de curso indicado y devuelve como resultado un nuevo vector con los DNI de las personas que estaban inscritas en ese curso (su tamaño debe coincidir con la cantidad de personas inscritas en el curso cancelado). La capacidad del vector de inscripciones no debe variar.

Si el coste temporal de este método es mayor de $\mathcal{O}(n)$, donde n es el número de inscripciones (valor del atributo `ocupados`), la nota máxima de este apartado será de 1 punto.