PRG - ETSInf. TEORIA. Curs 2018-19. Parcial 2.
3 de juny de 2019. Duració: 2 hores.

Nota: L'examen s'avalua sobre 10 punts, però el seu pes específic en la nota final de PRG és de **3 punts**.

1. 3 punts **Es demana:** escriure un mètode estàtic `void` de nom `sumInt`, que reba com a paràmetres un `String fileIn` amb el nom d'un fitxer de text del sistema, i un `String fileOut`. Se suposa que `fileIn` conté una seqüència d'enters, i el mètode ha d'escriure en `fileOut`, línia a línia els valors llegits de `fileIn`, i al final la suma de tots els valors llegits. En el cas en què es produïsca alguna excepció en llegir un `int`, s'ha d'escriure una línia amb el format `(Error:` *token incorrecte*`)`. Per exemple, si `fileIn` és un fitxer de text amb les següents dades:

```
4  5
20  1  2x3  10
3
```

llavors el fitxer de text resultant ha de contenir

```
4
5
20
1
(Error: 2x3)
10
3
Suma: 43
```

Si algun dels fitxers no es poguera obrir, el mètode ha de limitar-se a propagar l'excepció (comprovada) corresponent.

Solució:

```
public static void sumInt(String fileIn, String fileOut) throws FileNotFoundException {
     File fI = new File(fileIn), fO = new File(fileOut);
     Scanner in = new Scanner(fI); PrintWriter out = new PrintWriter(fO);
     int sum = 0;
     while (in.hasNext()) {
         try {
             int n = in.nextInt();
             out.println(n);
             sum += n;
         } catch (InputMismatchException e) {
             out.println("(Error: " + in.next() + ")");
         }
     }
    out.println("Suma: " + sum);
    in.close(); out.close();
}
```

2. 3.5 punts **Es demana:** afegir un mètode a la classe `QueueIntLinked` amb perfil:

```
public void split(int x)
```

tal que, donat un enter `x`, cerque la primera ocurrència de l'element `x` en la cua i el substituïsca pel parell d'elements `x / 2` i `x / 2 + x % 2`, un a continuació de l'altre. Si `x` no apareix, la cua no ha de canviar.

Per exemple, si s'invoca al mètode `q.split(9)` sent `q` la cua de longitud 6 $\leftarrow$ `1 -2 9 8 -3 5` $\leftarrow$, llavors `q` passa a ser la cua de longitud 7 $\leftarrow$ `1 -2 4 5 8 -3 5` $\leftarrow$.

IMPORTANT: En la solució només es pot accedir als atributs de la classe, quedant prohibit accedir als seus mètodes.

Solució:

```
public void split(int x) {
     NodeInt aux = this.first;
     while (aux != null && aux.data != x) {
        aux = aux.next;
     }
     if (aux != null) {
         aux.data = x / 2;
         aux.next = new NodeInt( x / 2 + x % 2, aux.next);
         if (aux == this.last) { this.last = aux.next; }
         this.size++;
     }
}
```

3. 3.5 punts **Es demana:** implementar un mètode estàtic `compress` tal que, donada una `ListPIIntLinked l` de la qual se suposa que els seus elements valen tots 0 o 1, retorne una altra llista de grandària aproximadament la meitat, tal que els seus elements representen als de `l`, prenent-los de dos en dos i d'esquerra a dreta. Així, on en `l` apareix una parella d'elements seguits `e1 e2`, en la nova llista apareix `e1 * 2 + e2` (un valor `0`, `1`, `2`, o `3` per a les parelles `00`, `01`, `10`, `11`, respectivament). En el cas en què en `l` quedara al final un element `e` desemparellat, en la nova llista apareixeria al final `e - 2` (un valor `-1` o `-2`). Els elements de la llista `l` no han de canviar, encara que la posició del punt d'interés sí que pot canviar.

   Per exemple, si `l` és una llista amb els elements `0 0 0 1 1 0 1 1 0 0 1`, el mètode ha de retornar una altra llista amb els elements `0 1 2 3 0 -1`.

   IMPORTANT: Se suposarà que el mètode s'implementa en una classe diferent a `ListPIIntLinked`, per tant, només es podran usar els mètodes públics de la classe.

Solució:

```
    /** Precondició: els elements de l valen 0 o 1. */
    public static ListPIIntLinked compress(ListPIIntLinked l) {
        ListPIIntLinked result = new ListPIIntLinked();
        int n = l.size();
        l.begin();
        while (n >= 2) {
            int e1 = l.get(); l.next();
            int e2 = l.get(); l.next();
            result.insert(e1 * 2 + e2);
            n = n - 2;
        }
        if (n == 1) { result.insert(l.get() - 2); }
        return result;
    }
```

ANNEX

Atributs de la classe `QueueIntLinked` i mètodes de la classe `ListPIIntLinked`.

```
public class QueueIntLinked {
    private NodeInt first, last;
    private int size;
    ...


}
```

```
public class ListPIIntLinked {
    ...
    public ListPIIntLinked() { ... }
    public void begin() { ... }
    public void insert(int x) { ... }
    public int get() { ... }
    public int remove() { ... }
    public void next() { ... }
    public int size() { ... }
    public boolean empty() { ... }
    public boolean isEnd() { ... }
}
```