Computer Programming – Theory – ETSINF – Academic year 2018/19
Second mid term exam – June 3^{rd}, 2019 – Duration 2 hours

Note: The maximum mark of this exam is 10 points, but its weight in the final grade of this subject is **3 points**.

1. 3 points Write an static method named `sumInt()` that does not returns any value of any type, so that it should be `void`. This method should have two parameters, more precisely two objects of the class `String`, `fileIn` and `fileOut`, containing the names of two files. The fist one, i.e. `fileIn`, should be the name of an existing file and it is assumed that such file contains a sequence of integers. The method should write on `fileOut` all the integers contained in `fileIn`, one value per line. And at the end the sum of all the read values. If any exception is thrown while the method tries to read integers from `fileIn`, a message with the format `(Error: <the invalid token> )` must also be printed on the output file, but the method should continue reading until reaching the end of file. For instance, if the input text file contains:

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

then the output text file should contain:

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

If any of the two files cannot be opened, the method should propagate, i.e. ignore, the corresponding checked exception.

Solution:

```
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("Sum: " + sum);
    in.close(); out.close();
}
```

2. 3.5 points In this exercise you have to add a new method to the class `QueueIntLinked` with the following profile:

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

such that, given an integer `x`, the new method searches the first occurrence of `x` inside the queue and substitutes it by two new elements with values `x/2` and `x/2+x%2`, one after the other. If `x` is not in the queue no changes are performed.

For instance, if the method is invoked as `q.split(9)` and the contents of `q` is ← `1 -2 9 8 -3 5` ←, then the contents of `q` becomes: ← `1 -2 4 5 8 -3 5` ←. Notice that the length of `q` was 6 and after the execution of the method is 7.

IMPORTANT: The solution must be implemented by using the attributes of the class `QueueIntLinked`, using other methods of the class is not allowed.

Solution:

```
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 points Write an static method named `compress()` with one object of the class `ListIntLinked` as the unique parameter. It is supposed that all the values contained in the list are zeroes or ones. The method must return a new list of integers, i.e. an object of the class `ListIntLinked`, whose components will be a representation of the elements of the input list taken in pairs from left to right. Given any pair of consecutive values in the input list, $e1$ and $e2$ respectively, in the output list it must be stored the value resulting of the expression $2 * e1 + e2$. The values stored in the new list can be any of $\{0, 1, 2, 3\}$ corresponding to the pairs $\{00, 01, 10, 11\}$. Additionally, if the length of the input queue is an odd number, then the last impaired value must be stored in the output list as $e - 2$. So, in such case the last value of the output list will be -1 or -2. The contents of the input list must remain untouched. The cursor of the input list can be changed.

For instance, if the input list contains `0 0 0 1 1 0 1 1 0 0 1`, the method must return a new list, i.e. the output list, with the following contents: `0 1 2 3 0 -1`.

IMPORTANT: The method must be implemented as an static method of a class different from `ListIntLinked`, so, the required method can only use the public methods of the class `ListIntLinked`.

Solution:

```
/** Precondition: all the elements in l are zeroes or ones. */
public static ListIntLinked compress( ListIntLinked l )
{
    ListIntLinked result = new ListIntLinked();
    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;
}
```

Appendix

Attributes of the class `QueueIntLinked` and methods of the class `ListIntLinked`:

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




}
```

```
public class ListIntLinked {
    ...
    public ListIntLinked() { ... }
    public void begin() { ... }
    public void next() { ... }
    public int get() { ... }
    public void insert(int x) { ... }
    public void append(int x) { ... }
    public int remove() { ... }
    public int size() { ... }
    public boolean empty() { ... }
    public boolean isValid() { ... }
}
```