COMP4040.G54PAD - Third Assignment
COMP4019.G54AAD - Lab Session 8
Fibonacci Heaps

In this assignment, your task is to implement the Fibonacci Heap data structure and its methods.

You can choose the programming language in which to complete the project among: Haskell, Python, C/C++, Java. Your code should run on the School's Linux servers.

1 Wheels

Implement *wheels*, that is doubly-linked circular lists. The following operations must be programmed so they have constant (amortized) cost:

- `emptyW`: the empty wheel, and `isEmptyW`: the Boolean test for emptiness;
- `rightW` and `leftW`: moving the head clockwise and anti-clockwise, respectively;
- `headW`: reading the element under the head;
- `insertW`: inserting a new element just to left of the head, it will be the new head element;
- `extractW`: delete and return the head element, the new head will be the next element on the right.

Your implementation should be abstract in the type of elements of the wheel: it must be capable of been instantiated with different types for elements.

2 Fibonacci Heaps

Implement *Fibonacci Heaps* as *recursive wheels*: a heap is a wheel of root elements, each of which is recursively linked to a sub-heap; also put into the node the degree of each root element, that is, the length of the wheel of its sub-heap.

Implement the following operations on Fibonacci Heaps in a *lazy* way, so they have constant cost.

- `emptyH`: the heap with no elements;
- `isEmptyH`: the Boolean test for emptiness;
- `insertH`: adding one element to an existing heap;
- `minimumH`: reading the minimum element from a heap;

3 Extraction and Consolidation

Implement `extractH` that deletes and returns the minimum element of a heap, following the description in the lecture:

- Remove the minimum from the root wheel;
- Concatenate the sub-heap of the minimum (which is now orphan) to the main root wheel;
- Consolidate the heap.

Implement `consolidate` in the way explained in the lecture, so every element of the root wheel has a different degree.

4 Testing and Marking

The main program in your implementation should be called `heapRun`: it must take as input the name of a text file containing a sequence of heap operations (similarly to what you have done for red-black trees), each consisting of the name of the operation (`minimum`, `insert`, `extract`). The `insert` operations are followed by an integer to be inserted. On execution it must output the results of all the minimum operations. (It should not output the elements obtained by the extraction operations).

You can test it on the file *heap0.txt* given on the course web page. If your implementation is correct, you should get this output:

```
minimum: 11
minimum: 56
minimum: 66
minimum: 74
minimum: 85
minimum: 86
minimum: 89
minimum: 91
minimum: 94
minimum: 100
```

Your submission will be marked according to the following criteria:

Wheels	20%
Fibonacci Heaps definition, `emptyH` and `isEmptyH`	20%
`insertH` and `minimumH`	20%
`extractH` and `consolidate`	30%
Style	10%

Submission: Submit an archive file (extension `.zip`, `.gz`, or `.gzip`) containing the files of your implementation of Fibonacci Heaps plus a text file `instructions.txt`, with **your name and ID number at the top**, describing how to compile and run it. The main program must be called `heapRun`.

The file name should be: `NameSurname_aad_assignment3`

For example, mine would be: `VenanzioCapretta_aad_assignment3.zip`