More Assertions in Eiffel

- Pre-conditions, post-conditions, and class invariants
- Invariants and variants of loops

```eiffel
print_array (arr: ARRAY [INTEGER])
-- print the values in the 'arr'
local
  i1: INTEGER
do
  from i1 := 0
  invariant in_range: i1 >= 0 and i1 < arr.count + 1
  variant arr.count - i1 + 1
  until i1 = arr.count
  loop
    io.put_integer (arr[arr.lower + i1])
    i1 := i1 + 1
  end
end
-- 1. {True} INIT {INV}
-- 2. {True} INIT {VAR >= 0}
-- 3. {INV and then not EXIT} BODY {INV}
-- 4. {INV and then not EXIT and then (VAR = v)} BODY {0 <= VAR < v}
```

- Checks

```eiffel
check positive_balance: balance > 0 end
```

Doubly-linked List in Eiffel

Oct. 13, 2009

In computer science, a doubly-linked list is a linked data structure that consists of a set of data records, each having two special link fields that contain references to the previous and to the next record in the sequence. It can be viewed as two singly-linked lists formed from the same data items, in two opposite orders.

![Doubly-linked List Diagram](image)

Design and implement your Eiffel class representing this data structure, with value type of INTEGER.

- Basic operations like creating a new list, inserting a node into the list, deleting a node from the list and sorting the nodes in the list (in descending or ascending order of their values) should be available.
- Make the contracts of your classes as complete as possible.
- Specify the variants and invariants in your loops.
- Test your classes