We wish to store a dynamic collection of records, each of the form \{timestamp, value\}, where value is a real number. The collection should support the operations `append_newer(t, v)` to add a new record (which we can assume has a larger timestamp than any existing record), `pop_oldest()` to remove the oldest record, and `get_oldest()` to inspect the oldest without removing it.

(a) Define the `Queue` abstract data type. Describe an implementation using a linked list. Explain how to use it for this dynamic collection of records. [3 marks]

The collection should also support `get_max()`, which returns a pointer to the record with the highest value in the collection. Ties may be broken arbitrarily.

(b) A simple implementation of `get_max()` simply scans through the entire list. What is the worst-case cost, given the number \(n\) of items in the collection? [1 mark]

(c) An engineer friend suggests keeping a pointer `maxrecord` to the record with the largest value so that the entire list only need be rescanned when the item pointed to by `maxrecord` is removed. Give an example to show that \(n\) operations could take \(\Omega(n^2)\) time. [3 marks]

(d) Explain the terms `amortized cost` and `potential method`. Explain the relationship between aggregate true costs and aggregate amortized costs. [4 marks]

(e) Devise an implementation in which all operations have \(O(1)\) amortized cost, and use the potential method to justify your answer. Illustrate what happens when we start with a list of values \([5, 8, 3, 6, 2]\) where 5 is oldest and 2 is newest, and then append a newer record with value 7. [Hint: Where is the largest item newer than `maxrecord`, and the largest item newer than this, and so on?] [9 marks]