We have a 2-player game in which players A and B take turns to remove either the leftmost or rightmost coin from of a row of coins of varying values. When no coins are left, the player with the higher total value wins.

Example: For a row of coins with values given by the list [20, 40, 30, 15], player A must select the rightmost coin (with value 15) in order to win with a total amount of 55, leaving player B with a total amount of 50.

(a) You are given three helper functions. The first function, poplast, takes a list and returns the list without its last element. The second function, last, takes a list of integers and returns the last value of that list. The third function, max, takes two integers and returns the larger of the two values.

Using these helper functions, write a recursive function winning_diff that takes a list of integers (representing the row of coins), and that returns the final difference of amounts between players A and B, assuming that player A goes first, and that player B plays optimally. If the difference is positive, player A wins, if it is negative, B wins. [8 marks]

(b) We are interested in implementing a functional deque that computes poplast and last in amortised constant time and that also enables access to the first element in amortised constant time. Write the code for the data type, and functions poplast and last. You may also need to code a function norm that guarantees amortised constant time in all circumstances. [8 marks]

(c) Consider the complexity of your algorithm for winning_diff for Part (a). For this question, assume that the three helper functions compute in constant time.

(i) Give the recurrence relation $T(n)$ for the running time of your algorithm, where $n$ is the number of coins in the row.

(ii) State the complexity of the algorithm in $O$-notation. [4 marks]