X6: Genome Rearrangements

Please submit your answers (as an *.rkt* file) via e-mail within a week (Subject should be: DCB X6). Keep in mind that the file must run without errors, and any procedures (names, arguments etc.) specified in a given task must be maintained.

Note 1: for each successfully completed lab-report you get a BONUS point that will be added on the final written examination points you will score.

Note 2: for each task, group your procedures, label them with a comment or two and provide a running example.

Task 1

```
ImprovedBreakpointReversalSort(π)

1 while b(π) > 0

2    if π has a decreasing strip

3          Among all possible reversals, choose reversal ρ that minimizes b(πρ)

4    else

5          Choose a reversal ρ that flips an increasing strip in π

6    π ← πρ

7    output π

8 return
```

The pseudo-code for *ImprovedBreakpointReversalSort* above leaves out a number of implementation details. Write procedures that perform the tasks of

- Line 1:
  Procedure:
  `Breakpoints: list -> number`

  Example:
  ```
  (Breakpoints '(0 3 4 6 5 8 1 7 2 9))
  ; Value: 7
  ```

- Line 6:
  Procedure:
  `Reverse: list, number, number -> list`

  Example:
  ```
  (Reverse '(1 2 3 4) 2 3)
  ```

```
;Value: (1 3 2 4)
(Reverse '(3 4 5 1 2 6) 1 4)
;Value: (1 5 4 3 2 6)
```

Note: The numbers are the positions for the reversal. The reversal should be done for all elements between these two positions (including the start and end position).

- Line 2:
  Procedure:
  ```
  HasDecrStrip: list -> boolean
  ```
  Example:
  ```
  (HasDecrStrip '(1 2 3 6 5))
  ;Value: #t
  ```
- Line 5:
  Procedure:
  ```
  FlipIncreasingStrip: list -> number x number
  ```
  Example:
  ```
  (FlipIncreasingStrip '(0 2 3 1 4 5))
  ;Value: (2 . 3)
  (FlipIncreasingStrip '(0 3 4 5 1 2 6))
  ;Value: (2 . 4)
  ```

Task 2

a) Use the procedures from Task 1 to implement an *ImprovedBreakpointReversalSort* procedure. You may use the implemented procedure *FindBestReversal* for line 3:

   Procedure:
   ```
   FindBestReversal: list -> number x number
   ```
   Example:
   ```
   (FindBestReversal '(0 3 4 1 2 5))
   ;Value: (2 . 4)
   ```
   The *FindBestReversal* is in the file FindBestReversal.txt together with a draft for the *ImprovedBreakpointReversalSort* procedure.

   Example:
   ```
   (ImprovedBreakpointReversalSort '(6 1 2 3 4 5))
   ; (0 5 4 3 2 1 6 7) ; (0 1 2 3 4 5 6 7)
   ; Done.
   ```
b) Run the *ImprovedBreakpointReversalSort* algorithm with π = 3 4 6 5 8 1 7 2. Show all reversals in the solution.

c) The if-test on line 2 ensures that the algorithm never gets stuck in a situation were there is no reversal that decreases the number of breakpoints. Can you construct a permutation *σ* where this if-test is needed (i.e. with no decreasing strips and no reversal that reduces the number of breakpoints)?

d) Since this is an approximation algorithm, there might be a sequence of reversals that is shorter than the one found by *ImprovedBreakpointReversalSort*. Can you construct a permutation *σ* for which it is the case that there is a shorter path?