CS2121/9643 – Assignment 3
due Mar. 29, 2016 (latest to submit: Apr. 1)

1. (10pt) Give an $\mathcal{O}(1)$-time algorithm for concatenating two doubly linked lists, each having head and tail pointers, such that the obtained doubly linked list has also head and tail. Explain why your algorithm runs in $\mathcal{O}(1)$-time.

2. (15pt) Give a linear-time recursive algorithm that finds the minimum and maximum values in an array of integers *without using any loops*.

3. (15pt) Give a recursive algorithm, *without using any loops*, that checks whether a string is a palindrome, that is, it is equal to its reverse, e.g., `racecar`.

4. (20pt) Give a concrete example of a hash table that is not full yet an insertion cannot be performed. You need to specify your example in all details:

   - the hash table size,
   - the hash function used,
   - the current content of the hash table,
   - the element to be inserted, and
   - why this is not possible.

5. (40pt) You are required to implement and compare linear probing and double hashing. You are given four files:

   `HashTable_DoubleHashing.py`
   `HashTable_LinearProbing.py`
   `my_array.py`
   `testHashing.py`

   The first two contain classes implementing linear probing and double hashing hash tables. You need to fill in the code for the searching function `_findSlot()`. The last one contains the comparison. A prime `size` is given for the hash table size and random `data` is generated to be inserted in the tables. Two hash tables, `h_lp` and `h_dh`, are created and you need to compare their behaviour with respect of the average number of slots accessed, given five different load factors: 0.25, 0.5, 0.67, 0.80, 0.95. Further help on how to perform the comparison is included in the `testHashing.py` file.

   The output should look like this:

```
average slots accessed:

load =  0.25
linear probing:     1.15
double hashing:     1.09

load =  0.5
linear probing:     2.02
double hashing:     1.53

load =  0.67
linear probing:     4.46
double hashing:     2.37
```

```
load =  0.8
linear probing:     12.82
double hashing:     4.29

load =  0.95
linear probing:     233.42
double hashing:     21.19
```

The values you obtain can be slightly different from the above.

Note: Submit your solution on `owl.uwo.ca`: .py files for question 5 and a .pdf file with all the remaining answers. Do not submit Python code for questions other than 5! Present your algorithms in pseudocode with plain English description.