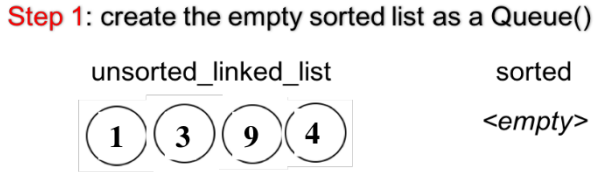
APS106 - Lab #9

This lab will test your ability to define classes and methods for linked data structures as well as use them in the implementation of a sorting algorithm. Place appropriate comments in your program. ***Due: 11:59pm, Friday, April 5th, 2019***

Question: A variation of **Selection sort** algorithms divides the input list into two parts: a sublist of items already sorted and a sublist of items remaining to be sorted. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the largest (i.e. in descending order) element in the unsorted sublist, putting it in sorted list, and repeating the process.

Here's an easy step-by-step process for understanding this algorithm using the functions provided:

Step 1: create the empty sorted list as a Queue()

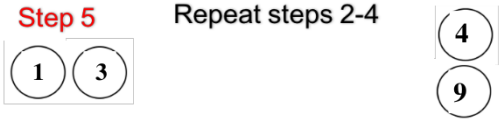


Step 2: maxvalue() function finds that 2 as the smallest element

Step 3: append() function appends 2 in the sorted list

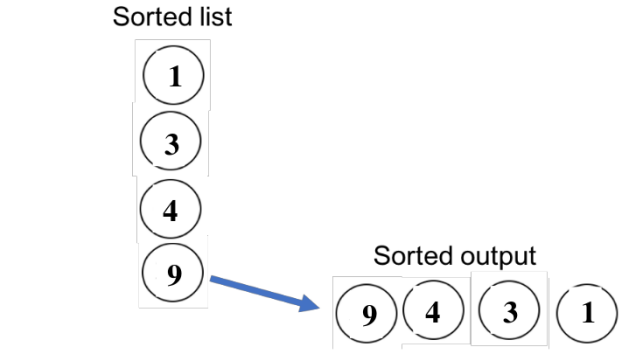
Step 4: remove_max() removes 2 from unsorted_linked_list And places it in queue

Step 5 Repeat steps 2-4



Step 6 Keep repeating steps 2-4 till unsorted_linked_list is empty and queue is full!

Step 7: Concatenate the sorted list from the queue



Below is a pseudocode for the *selectionSort()* function you will complete in your script (It uses recursion!):

```
selectionSort

Input: unsorted_list
Output: a copy of the input list sorted in ascending order

if the length of the unsorted_linked_list <= 1: #nothing to sort, the
    return unsorted_linked_list                 #list has 0 or 1 items
else:
    sorted_list <- Queue()
    max_element <- max(unsorted_list)

    append max_element to the sorted_list
    remove max value from the unsorted_list
     #recursion happens below
    concatenate sorted_list and selectionSort(unsorted_linked_list))

    return sorted
```

Here is an example of this sort algorithm sorting a list of five elements recursively. Recursion results in *concatenated_list* concatenating elements in descending order:

```
unsorted_list is 64 25 12 22 11 #this is the initial state of the list to sort
sorted list <64> unsorted list <25, 12, 22, 11>
sorted list <25> unsorted list <12, 22, 11>
sorted list <22> unsorted list <12, 11>
sorted list <12> unsorted list <11>

concatenated list <12, 11>
concatenated list <22, 12, 11>
concatenated list <25, 22, 12, 11>
concatenated list <64, 25, 22, 12, 11>
```

Sample Inputs and Outputs

```
>>> print(selectionSort(Queue([5,2,1,3])))
[5, 3, 2, 1]

>>> print(selectionSort(Queue([2,8,7,9,3,6,0])))
[9, 8, 7, 6, 3, 2, 0]
```

TO DO:
Download the file lab9.py, follow the directions on the script for completing the functions and upload your version of lab9.py to MarkUs. You can make use of all the functions provided.