50.004 ISTD @ SUTD Algorithms class Fall 2017
Dr. Alexander Binder

Please do the following before starting this exercise

- Install Python 3
- Download *arrtree3.py* and *drawtree3.py* to the same location.
- Create a separate file to test your code, named *testcode.py* for convention. If you are more comfortable with using the terminal feel free to do so.
- It is encouraged to have all the python files in the same file directory as the python file you use to test code imports code from *arrtree3.py* and *drawtree3.py*.

For all code that needs to be implemented, please write it in *arrtree3.py*, there will be instructions given to you there.

Task 1: Simulate max heapify by class member swapv.

1. Copy and paste this code to the file *testcode.py* and run it :

```
from arrtree3 import *           #loads the class definitions
v2 = [3,7,11,66,6,1,5,54,9]      #create an array
h2 = treesimple(v2)             #create a tree from the underlying array
vistree2(h2)                     #plot the tree
```

2. Simulate max heapify starting at node 1 (array index starts at zero) with key 7 (you do not need to code the function *max_heapify* yet). You can do this by calling the function *swapv(i,j)* in *testcode.py* to visualise the tree in the console output.

```
h2.swapv(i,j) #h2 is the tree and i,j are indices of nodes to be swapped.
```

   If you do a wrong swapv, you can call the function again on the same node indices to resolve this issue.

3. Then do max-heapify for node 0 with key 3 for the h2.

4. Try to use swapv to perform max-heapify on nodes 1 and 0 of another array:
`v3=[3,7,5,9,6,1,11,54,66]`

5. After calling swapv on the array in part 4, do you need to do anymore swapping? Hint: Consider the heap property for max heaps.

Task 2: Implement max_heapify and build_max_heap for the treesimple class

1. Implement *max_heapify* method in *arrtree3.py*
2. Implement *build_max_heap* method in *arrtree3.py*

Hint: Please refer to the code in *arrtree3.py* for helpful hints. More instructions can be found within the code of *arrtree3.py*.

Task 3: Checking code for errors

1. Use the pseudo-code and your pencil and paper to visualise the process of max heapify and build max heap.

2. Run the codes you wrote for *max_heapify* and *build_max_heap* inside *arrtree.py* and run some code in *testcode.py* to see if the result obtained by the code is correct. One way you can do this is through calling the function

   ```
   h3 = treesimple(v3)
   max_heapify3(h2,1)
   build_max_heap(h3)
   ```

   inside the file *testcode.py*. Of course using the terminal is another viable option, if you're up for it.

3. Run 2 iterations of step 1 and 2 for different input arrays, it can be any array you want. If your code gives an incorrect result, look back to your pseudo code and visualisations to identify what operations you need to implement for your methods in *arrtree.py* .

Task 4: Code a heap sort and extract max functions.

1. Observe the relationship between extract max and a heap sort.

2. Try coding an extract max function, followed by heap sort in *arrtree3.py*

Hint: What operations do you need for an extract max? Note that you will need to run max heapify after you have decreased the heap counter. Task 3 can help you get an idea of how to implement the code.