This assignment consists of a written component and a programming component. Please read the course website carefully to ensure that you submit each component correctly.

**Written component**

**W1. [7 marks]** Write pseudocode for an algorithm that finds the minimum value in a max-heap. Use an array based implementation, in which the $n$ items contained in the heap are located in the elements 0 through $n$-1 of the array $a$. The value of $n$ is stored as a variable, to which you have access.

Your algorithm should be faster (although not necessarily asymptotically faster) than simply scanning every item in the heap.

What is the worst-case running time of your algorithm (in asymptotic notation)? Justify your answer.

**W2. [7 marks]** Write pseudocode for an algorithm that takes as input a max-heap (using an array based implementation) and an integer key, and removes the key from the heap (if present; otherwise, the heap is unchanged). The $n$ items contained in the heap are located in the elements 0 through $n$-1 of the array $a$. The value of $n$ is stored as a variable, to which you have access.

What is the worst-case running time of your algorithm (in asymptotic notation)? Justify your answer.

**W3. [6 marks]**

a) Many of the data structures we have discussed are efficient for (or even optimized for) certain operations, while performing worse for others. Complete the table below, giving the worst case running time for each operation in asymptotic notation. Some of these boxes you will be able to fill in directly from your notes/slides/text; others will require some thought. For the boxes marked with an asterisk (*), you must briefly describe an algorithm to implement the operation, and justify your running time. (Full pseudocode is not required, but your algorithm and analysis must be clear from your description.) In each case, use $n$ to denote the number of items in the structure, and $m$ to denote the size of the array (for array-based structures), where $n \leq m$. 
<table>
<thead>
<tr>
<th></th>
<th>Find Minimum key</th>
<th>Find Maximum Key</th>
<th>Search for a key</th>
<th>Insert a Key</th>
<th>Remove an arbitrary key</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVL Tree (linked node implementation)</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max-Heap (array implementation)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash table (open addressing, with load factor less than 50%)</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Find minimum key in an AVL tree (algorithm/running time):

Find the minimum value in a hash-table with open addressing (algorithm/running time):

b) Given your completed table for part a), consider a workload where we will often need to find the minimum and maximum value, rarely have to search for a value, frequently add values, but rarely remove values. Which of these data structures is the best choice for such a workload? Justify your answer.

Programming component
Please read the course website carefully to ensure that you are using the correct version of Python and the correct style.

P1. [20 marks] For this question, you will implement a hash table making use of open addressing. This table will store key-value pairs, allowing access by key (i.e., like a map/dictionary).

The interface for the OAHash class is provided in the file oahash.py, which accompanies this assignment. Implement each of the functions that do not have implementations (i.e., those with ‘pass’ as bodies); a handful of functions have been implemented for you (e.g., __str__()), to help you in your work. You may modify the string functions if you find it beneficial during development/debugging.
You have been provided with two additional files. **hashentry.py contains the HashEntry class, which you must use to store data in your hash table.** (Key-value pairs are placed in HashEntries, which are then stored directly in the array.) myarray.py contains the Myarray class, which we covered earlier in the course. **You must use a Myarray object as the array in your Hash Table. Do not use a Python list, or any other structure.**

You may add functions and variables to the OAHash class as necessary, but do not modify the signatures provided for any functions.

Your implementation must accept strings as keys. The hash function should take a string as input, and generate a non-negative integer hash code by: a) summing the Unicode values of each character in the sequence; b) returning the sum modulo the size of the array. You can find the Unicode value of a character using the built-in ord() function.

Note: Several of the functions require you to return tuples. You might not be familiar with returning tuples (and receiving returned tuples). To return a tuple in Python, simply return multiple values, separated by commas. For example, the following function takes a number, and returns a tuple consisting of the input value squared, and cubed.

```python
def tuple_test(x):
    return x**2, x**3
```

If you simply call the function in the interpreter, or store its return value in a variable, the result is a tuple:

```python
>>> tuple_test(2)
(4, 8)
>>> r = tuple_test(3)
>>> r
(9, 27)
```

However, if you assign the function result to multiple variables, each variable will be assigned the corresponding value from the tuple:

```python
>>> square, cube = tuple_test(4)
>>> square
16
>>> cube
64
```

**A capture tracing interactive simple usage and correct operation of the OAHash class:**

```python
>>> h = OAHash(6,1)
>>> ord('A')
65
>>> h.hash('A')
5
>>> h.add_item('A', 1)
```
(A->1, [5])
>>> h.add_item('B', 2)
  (B->2, [0])
>>> h.add_item('C', 3)
  (C->3, [1])
>>> h
  0:B->2
  1:C->3
  2:None
  3:None
  4:None
  5:A->1

>>> h.add_item('G', 4)
  (G->4, [5, 0, 1, 2])
>>> h
  0:B->2
  1:C->3
  2:G->4
  3:None
  4:None
  5:A->1

>>> h.find('G')
  (G->4, [5, 0, 1, 2])
>>> h.find('H')
  (None, [0, 1, 2, 3])
>>> h.delete_item('A')
  (A->1, [5])
>>> h
  0:B->2
  1:C->3
  2:G->4
  3:None
  4:None
  5:None->None