Output:

```
Part 1, Section B - Creating only max heap
1. Heapify Swap Counter: 992
2. OneByOne Swap Counter: 8977

Part 2, Section B - Heap creation and sorting (Random -> HeapSort)
1. Heapify Swap Counter: 8083
2. OneByOne Swap Counter: 8518
```

For creating a max-heap only (no sorting), the average case

1. Heapify – always 992
2. One by one – always 8977

Therefore it is Theta (n).

For average case for heap creation and sorting –

1. Heapify – 8083
   Would also be Theta (n) because it is based on the same heapify method where all the if statements are O(1) and the swap method
2. One by one – 8518

Worst case scenario -

1. Heapify method was faster by about 500 steps
   - Heap creation: O(n) since heapify needs to go through each element to swap them
   - Heap sort: O(n * log n) since heap is already created and heapsort method is O(n * log n)
2. One by one method
   - Heap creation: O(n * log n) since subsequent elements were progressively larger and thus required more swaps to arrive at root
   - Heap sort: O(n * log n) since heap is already created and heapsort method is O(n * log n)