Computer Science I – Exercise
Heaps

1) In an array-based implementation of a Heap, the left-child of the left-child of the node at index i, if it exists, can be found at what array location?

$4i$

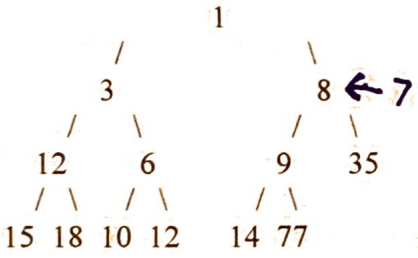
Child: $2i$ → $2i$ ⇒ $2(2i) = 4i$

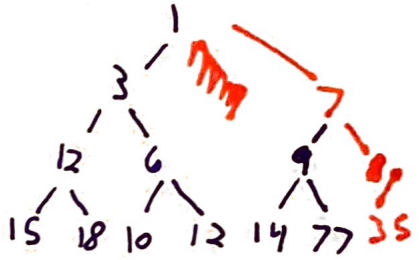
2) In an array-based implementation of a Heap, the right-child of the right-child of the node at index i, if it exists, can be found at what array location?

$4i+3$

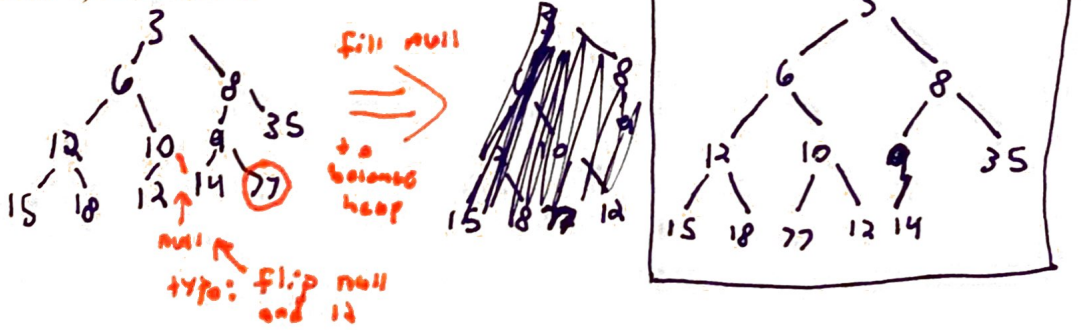
Child: $2i+1$ → $2i+1$ ⇒ $2(2i+1)+1 = 4i+2+1 = 4i+3$

3) Show the result of inserting the item 7 into the heap shown below:





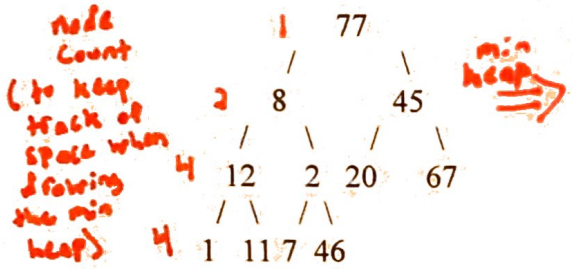
4) Show the result of removing the minimum element from the original heap in question #3 (without 7) from above.

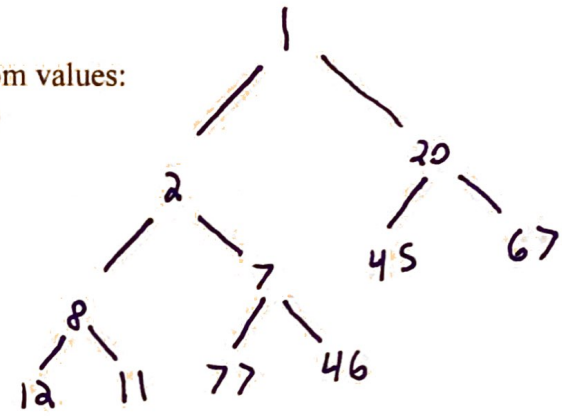


5) Show the array representation of the original heap from question #3

1 3 8 12 ~~10~~ 6 9 35 15 18 10 12 14 77

6) Run the whole Heapify function on the following random values:
(this is the function that builds a heap in O(n) time)





7) Explain each step shown in the code below, for the percolateDown function:

```
void percolateDown(struct heapStruct *h, int index) {

   int min; //Will be the minimum value

   if ((2*index+1) <= h->size) { // check if index is within bounds

      min = minimum(h->heaparray[2*index], 2*index, h->heaparray[2*index+1], 2*index+1);

      if (h->heaparray[index] > h->heaparray[min]) { //Check comparison of both index (node & min child)
         swap(h, index, min); //Swap if child is smaller
         percolateDown(h, min); // recursive call
      }
   }
   else if (h->size == 2*index) { // check if node only has one child
      if (h->heaparray[index] > h->heaparray[2*index]) // check if child is smaller
         swap(h, index, 2*index); //Swap node with child
   }
}
```

index of
// get minimum value of node's children

(Note: Please reference heap.c *without looking at this function*, if necessary.)