Name:_____

PID:_____

Computer Science I – Quiz – Sorting & Heaps

1) **(2pts)** What is the best case run time of Quick Sort? **O(n log n)**

2) **(2pts)** What is the worst case run time of Quick Sort? **O(n^2)**

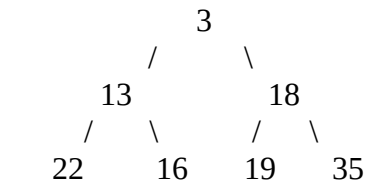
3) **(2pts)** What is the best case run time of Merge Sort? **O(n log n)**

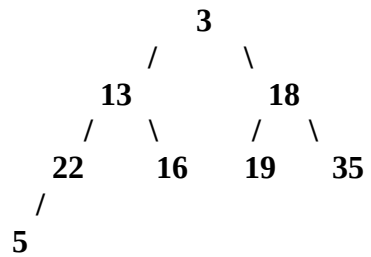
4) **(2pts)** What is the worst case run time of Merge Sort? **O(n log n)**

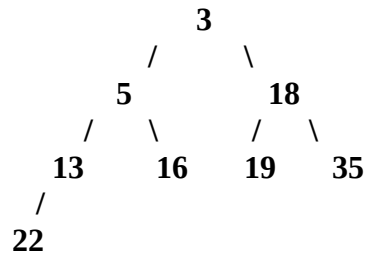
5) **(2pts)** What is the best case run time of Insertion Sort? **O(n)**

6) **(2pts)** What is the worst case run time of Insertion Sort? **O(n^2)**

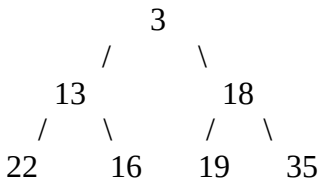
7) **(3 pts)** Show the result of inserting the item 5 into the heap shown below:

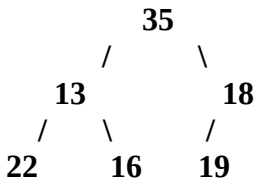


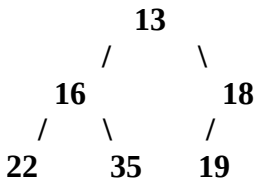




8) **(3 pts)** Show the result of removing the minimum element from the original heap in question #7 (without 5) from above.







9) **(2pts)** Show the array representation of the original heap from question #7.

index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**…**
value	X	3	13	18	22	16	19	35	X	…

10) **(Bonus 3pts)** Given the following specifications for a heap implementation, implement the heap sort functions below.

```
struct heapStruct {

    int* heaparray;
    int capacity;
    int size;
};

struct heapStruct *initHeap();
struct heapStruct *heapify(int *values, int length);
void percolateDown(struct heapStruct *h, int index);
void percolateUp(struct heapStruct *h, int index);
void insert(struct heapStruct *h, int value);
int removeMin(struct heapStruct *h);

void sort(int values[], int length) {


     struct heapStruct *myHeap = initHeap();
     myHeap = heapify(values, length);

     int i = 0;
     for (i = 0; i < length; i++) {

          values[i] = removeMin(myHeap);

     }

}
```