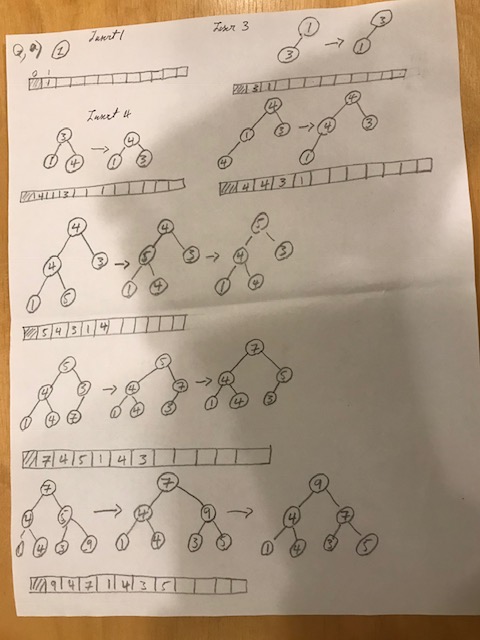
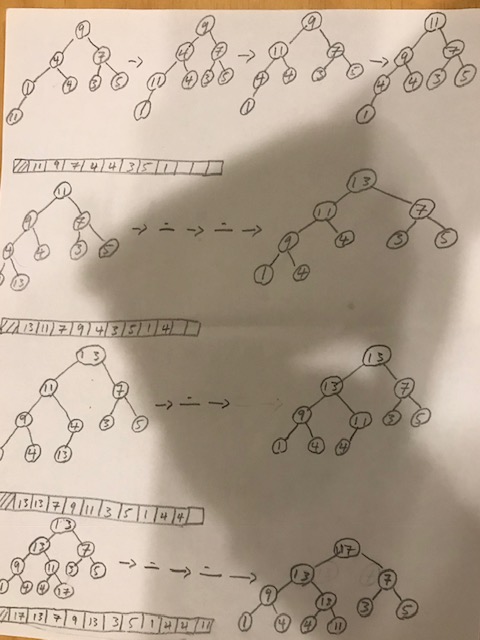
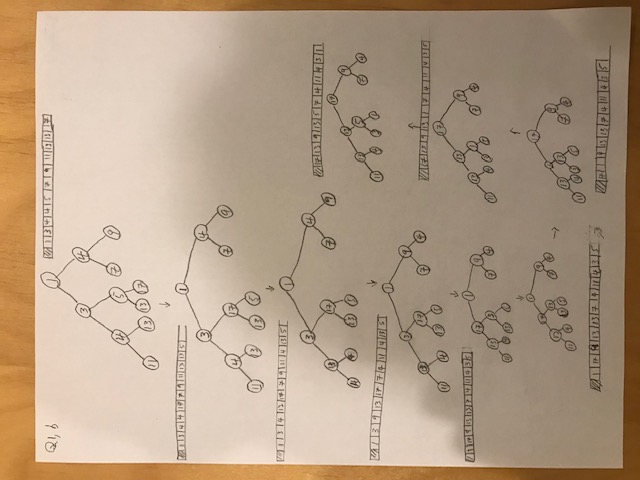
W3D2

AnswerQ1(a)





(b)



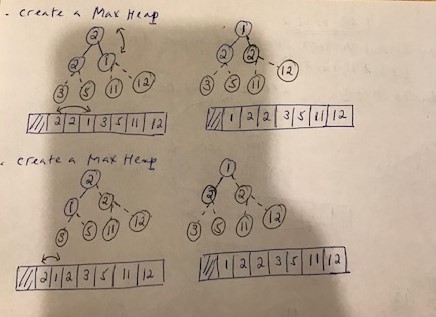
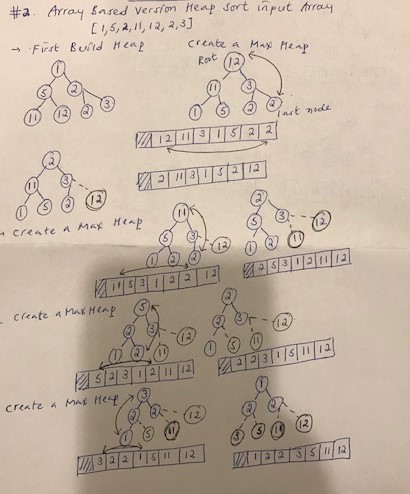
C,21 number of comparisons

D,16 number of comparisons

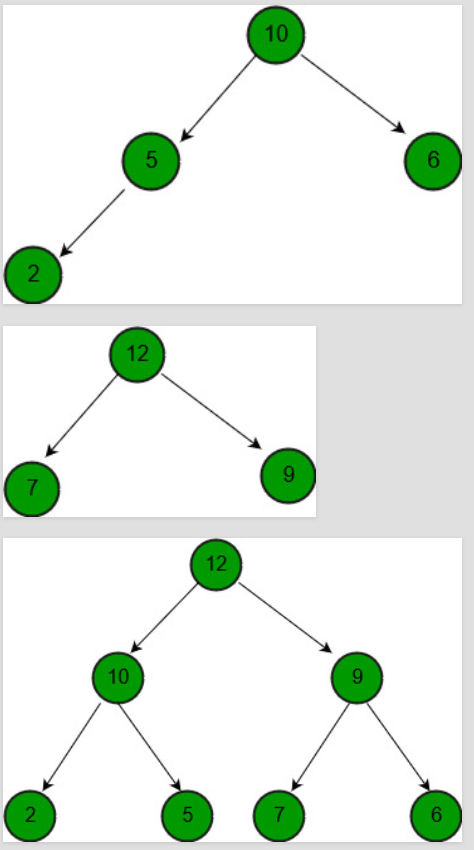
E,20 number of swaps

F,8 number of swaps

AnswerQ2



AnswerQ3



(a)

// Standard heapify function to heapify a

    // subtree rooted under idx. It assumes

    // that subtrees of node are already heapified.

    public static void maxHeapify(int[] arr, int n,

                                             int i)

    {

        // Find largest of node and its children

        if (i >= n) {

            return;

        }

        int l = i \* 2 + 1;

        int r = i \* 2 + 2;

        int max;

        if (l < n && arr[l] > arr[i]) {

            max = l;

        }

        else

            max = i;

        if (r < n && arr[r] > arr[max]) {

            max = r;

        }

        // Put maximum value at root and

        // recur for the child with the

        // maximum value

        if (max != i) {

            int temp = arr[max];

            arr[max] = arr[i];

            arr[i] = temp;

            maxHeapify(arr, n, max);

        }

    }

    // Merges max heaps a[] and b[] into merged[]

    public static void mergeHeaps(int[] arr, int[] a,

                                  int[] b, int n, int m)

    {

        for (int i = 0; i < n; i++) {

            arr[i] = a[i];

        }

        for (int i = 0; i < m; i++) {

            arr[n + i] = b[i];

        }

        n = n + m;

        // Builds a max heap of given arr[0..n-1]

        for (int i = n / 2 - 1; i >= 0; i--) {

            maxHeapify(arr, n, i);

        }

    }

// Driver Code

    public static void main(String[] args)

    {

        int[] a = {10, 5, 6, 2};

        int[] b = {12, 7, 9};

        int n = a.length;

        int m = b.length;

        int[] merged = new int[m + n];

        mergeHeaps(merged, a, b, n, m);

        for (int i = 0; i < m + n; i++)

            System.out.print(merged[i] + " ");

        System.out.println();

    }

1. Best case O(n)
2. Worst case O(n)
3. All is the same