Assignment_2

February 15, 2019

Question 1

```
In [4]: def swap(pos1,pos2,listToBeAltered):
            temp = listToBeAltered[pos1]
            listToBeAltered[pos1] = listToBeAltered[pos2]
            listToBeAltered[pos2] = temp

            pass



        def heapify(listToConvert, position):
            ##position form 1
            initParent = listToConvert[position]
            leftChild = -float("Inf")
            rightChild = -float("Inf")
            newPosition = position

            if 2*position+2 < len(listToConvert): ##index in range or there exists a child
                rightChild = listToConvert[2*position+2]

            if 2*position+1 < len(listToConvert):
                 ##pos
                leftChild = listToConvert[2*position+1] ##2*pos
                 ## 2*pos+1


                if  leftChild > initParent and   leftChild >rightChild: ##left child is the gr
                    swap(2*position+1,position,listToConvert) ##swap parent with right
                    newPosition = 2*position+1


                elif  rightChild > initParent and rightChild > leftChild: ##right child is the
                    swap(2*position+2,position,listToConvert) ##swap parent with right
                    newPosition = 2*position+2

                if newPosition != position:
                    print(listToConvert)

                    heapify(listToConvert, newPosition)


            pass


        def extractMax(heapedList):
            maxElem = heapedList[0]
            heapedList[0] = heapedList[-1]
            del heapedList[-1]
            heapify(heapedList,0)

            return maxElem

        def buildMaxHeap(listToConvert):
            ##from last but one layer of nodes do heapify
            lastPosition = len(listToConvert)-1

            for i in range(int(lastPosition/2 -1),0,-1):
                heapify(listToConvert,i)

        myList = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]


        extractMax(myList)
```

```
[13, 1, 9, 5, 12, 8, 7, 4, 0, 6, 2]
[13, 12, 9, 5, 1, 8, 7, 4, 0, 6, 2]
[13, 12, 9, 5, 6, 8, 7, 4, 0, 1, 2]
```

```
Out[4]: 15
```

Question 2 Similar to merge sort algorithm we can add another condition where while merging when the elements are equal just add one of it to the list and increment the pointer for both i and j.

```
In [6]: def mergeWithoutDuplicates(leftList, rightList):
            mergedList = []
            leftLen = len(leftList)
            rightLen = len(rightList)
            i = j = 0

            while(i < leftLen and j < rightLen):

                if (leftList[i] < rightList[j]):
```