Merge sort –module -9

ACTIVITY 2🡺NORMAL MERGE SORT TWO DOWN APPROACH

1. EASYQUESTION
2. EASYQU | ESTION
3. EA |SY|QU ES |TI |ON
4. AESY |QU EIST |ON
5. AEQSUY EINOST
6. AEEINOQSSTUY

Activity-4 🡺BOOTTOM UP APPROACH

1. EASYQUESTION
2. EA |SY|QU |ES |TI|ON
3. AE |SY|QU|ES|IT|NO
4. AESY |EQSU|INOT
5. AEEQSSUY |INOT
6. AEEINOQSSTUY

ACTIVITY-7

Give the sequence of subarray sizes in the merges performed by both the top-down and the bottom-up merge sort algorithms, for N = 39.

1. 19 | 20
2. 9 | 10 10 | 10
3. 4 | 5 | 5 | 5 5 | 5 | 5 | 5
4. 2 |2 2 |3 | 2|3 2|3 2 |3 | 2 | 3 | 2 |3 | 2 | 3
5. 2| 2 2| 2|1 | 2| 2|1 | 2| 2|1 2| 2|1 | 2|2 |1 | 2| 2|1 | 2| 2|1

Bottom –up🡺

1. 2 | 2| 2| 2| 2| 2| 2| 2 |2 |1 | 2| 2| 2| 2| 2| 2| 2| 2| 2| 2
2. 4 | 4| 4| 4| 3 | 4 | 4| 4| 4| 4
3. 8| 8 |3 | 8 | 8 | 4
4. 19 | 20
5. 39

**Activity8**

**write the missing statements in the below code:**

**public static void merge(Comparable[] a, int lo, int mid, int hi) {**

**int i = lo,**

**j = mid+1;**

**for (int k = lo; k <= hi; k++)**

**aux[k] = a[k];**

**for (int k = lo; k <= hi; k++)**

**if (i > -----------)                                         //Blank 1🡺 mid**

**a[k] = aux[--------];                      //Blank 2🡺 j++**

**else if (--------- > hi )                                //Blank 3 🡺j**

**a[k] = aux[----------];                   //Blank 4🡺 i++**

**else if (less(aux[--------], aux[--------]))   //Blank 5 //Blank 6🡺 j 🡺 i**

**a[k] = aux[j++];**

**else**

**a[k] = aux[i++];**

**}**