KIIT Deemed to be University
Online Mid Semester Examination (Autumn Semester-2020)

Design & Analysis of Algorithms (DAA)
(CS-2012)
Solution & Evaluation Scheme

Full Marks=20 **Time:1 Hour**

SECTION-A
(Answer All Questions. All questions carry 2 Marks)

Time:20 Minutes **(5×2=10 Marks)**

Question No	Question Type (MCQ/ SAT)	Question	Answer & Evaluation Scheme
Q.No: 1(a)	**SAT**	Rank the following functions by order of growth in increasing sequence? log √n, √n, 2^{2^n}, √n log n	**Answer** log √n, √n, √n log n, 2^{2^n} **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks
	SAT	Rank the following functions by order of growth in increasing sequence? log √n, n, 2^{n^2}, n² log n	**Answer** log √n, n, n^2 log n, 2^{n^2} **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks
	SAT	Rank the following functions by order of growth in increasing sequence? n^2, n^{2^n}, nlog n, 500	**Answer** 500, nlog n, n^2, n^{2^n} **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks
	SAT	Rank the following functions by order of growth in increasing sequence? n^{2^n}, nlog √n, n³, √n	**Answer** √n, nlog √n, n³, n^{2^n} **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks

Q.No: 1(b)

SAT

What is time complexity of the following function fun1()?

```
int fun1(int n)
{
   int i, j, s=0;
   for (i = n; i >= n; i /= 2)
         for (j = 0; j < i; j++)
               s += 1;
   return s;
}
```

Answer

$\Theta(n)$

Scheme

- Correct Answer : 2 marks
- Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks

SAT

What is time complexity of the following function fun2()?

```
int fun2(int n)
{
   int i, j, s=0;
   for (j = 0; j < n; j++)
       for (i = n; i>=n; i /= 2)
                 s += 1;
   return s;
}
```

Answer

$\Theta(n)$

Scheme

- Correct Answer : 2 marks
- Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks

SAT

What is time complexity of the following function fun3()?

```
int fun3(int n)
{
   int i, j, s=0;
   for (i = 1; i <= n; i ++)
         for (j = 1; j <= i; j++)
               s += 1;
   return s;
}
```

Answer

$\Theta(n^2)$

Scheme

- Correct Answer : 2 marks
- Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks

SAT

What is time complexity of the following function fun4()?

```
int fun4(int n)
{
   int i, j, s=0;
   for (i = n; i <= n; i ++)
         for (j = 1; j <= i; j++)
               s += 1;
   return s;
}
```

Answer

$\Theta(n)$

Scheme

- Correct Answer : 2 marks
- Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks

Q.No: 1(c)

SAT

What is the running time of QUICKSORT when all elements of array A have the same value?

Answer

It is equivalent to the worst case running of QUICK-SORT, that is $T(n)=\Theta(n^2)$

Scheme

- Correct Answer : 2 marks
- Wrong answer: 0 mark

SAT

What is the running time of INSERTION SORT when all elements of array A have the same value?

Answer

It is equivalent to the best case running of INSERTION-SORT, that is $T(n)=\Theta(n)$

Also $T(n)=\Theta(n\log n)$ is true. It is also equivalent to the Best case as it gives balanced partitioning (mostly the way we write the

			partitioning) running of QUICK-SORT, that is $T(n)=\Theta(n \log n)$ **Scheme** ● Correct Answer : 2 marks ● Wrong answer: 0 mark
	SAT	What is the running time of merge sort when all elements of array A have the same value?	**Answer** $T(n)=\Theta(n \log n)$ **Scheme** ● Correct Answer : 2 marks ● Wrong answer: 0 mark
	SAT	What is the nature of data set and position of pivot element, so that quick sort exhibits worst case behaviour.	**Answer** Sorted or reversely sorted data with first/last element is choosen as pivot element. Most appropriately, Sorted or reversely sorted data with the first/last element is chosen as pivot element. **Scheme** ● Correct Answer : 2 marks ● Wrong answer: 0 mark
Q.No: 1(d)	**SAT**	What is the effect of calling MIN-HEAPIFY(A, i) for i > size[A]/2?	**Answer** No effect. All nodes at index i > size[A]/2 are leaves. **Scheme** ● Correct Answer : 2 marks ● Wrong answer: 0 mark
	SAT	What is the effect of calling MAX-HEAPIFY(A, i) for i > size[A]/2?	**Answer** No effect. All nodes at index i > size[A]/2 are leaves. **Scheme** ● Correct Answer : 2 marks ● Wrong answer: 0 mark
	SAT	Where in a min-heap might the largest element reside, assuming that all elements are distinct?	**Answer** The largest element must be a leaf node. **Scheme** ● Correct Answer : 2 marks ● Wrong answer: 0 mark
	SAT	Where in a max-heap might the smallest element reside, assuming that all elements are distinct?	**Answer** The smallest element must be a leaf node. **Scheme** ● Correct Answer : 2 marks

			● Wrong answer: 0 mark
Q.No: 1(e)	**MCQ**	What is the solution to the recurrence $T(n) = 4T(n/2) + n^2$, $T(1)=1$ A) $T(n) = \Theta(n)$ B) $T(n) = \Theta(\log n)$ C) $T(n) = \Theta(n^2\log n)$ D) $T(n) = \Theta(n^2)$	**Answer** C **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks
	MCQ	What is the solution to the recurrence $T(n) = 16T(n/4) + n$, $T(1)=1$ A) $T(n) = \Theta(n)$ B) $T(n) = \Theta(\log n)$ C) $T(n) = \Theta(n^2\log n)$ D) $T(n) = \Theta(n^2)$	**Answer** D **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks
	MCQ	What is the solution to the recurrence $T(n) = 6T(n/4) + n^2\log n$, $T(1)=1$ A) $T(n) = \Theta(n)$ B) $T(n) = \Theta(\log n)$ C) $T(n) = \Theta(n^2\log n)$ D) $T(n) = \Theta(n^2)$	**Answer** C **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks
	MCQ	What is the solution to the recurrence $T(n) = 3T(n/3) + \sqrt{n}$, $T(1)=1$ A) $T(n) = \Theta(n)$ B) $T(n) = \Theta(\log n)$ C) $T(n) = \Theta(n^2\log n)$ D) $T(n) = \Theta(n^2)$	**Answer** A **Scheme** ● Correct Answer : 2 marks ● Incorrect answer, but some valid stpes/explanation: 0.5-1.5 Marks

SECTION-B

(Answer Any One Question. Each Question carries 10 Marks)

Time: 30 Minutes **(1×10=10 Marks)**

Question No	**Question**
Q.No: 2	Given a set S of n integers and another integer x, determine whether or not there exist two elements in S whose sum is exactly x. Describe a $\Theta(n\log n)$ time algorithm for the above problem. **Scheme** ● Correct $\Theta(n\log n)$ algorithm : 10 Marks ● Correct algorithm with other than $\Theta(n\log n)$: 4-7 Marks ● Algorithm approaches to solution (not fully correct) : 0.5-5 Marks

Answer

- Sort the set S using merge sort. Then for each y ∈ S separately use binary search to check if integer x − y exists in S. Sorting takes time Θ(nlogn). Binary search takes time O(logn) and is executed n times. The total time is thus Θ(nlogn).
- If S is sorted, the problem can also be solved in linear time by scanning the list S at the same time forward and backward directions:
- **Psedocode**

```
/*S is an array of n integers. lb & ub represent the array indices*/
SUM_SEARCH(S, n, x)
{
    MERGE-SORT(S, 1, n)
    lb ← 0          //Intial value of lb
    ub ← n - 1      //initial value of ub
    while (lb < ub)
    {
        if (S[lb] + S[ub] == x)
            return true
        else if (S[lb + S[ub] < x)
            lb ← lb + 1
        else
            ub ← ub -1
    }
    return false
}
```

Q.No: 3 Write HEAPIFY() procedure and derive its time complexity. The elements of a heap structure are given as < 21, 1, 17, 8, 9, 6, 7, 4, 3, 8, 5 >. Find the node i, where the procedure HEAPIFY(i) should be applied to covert the given sequence into a max-heap. Show all the steps for performing HEAPIFY(i) operation on the above sequence.

Scheme

- MAX-HEAPIFY Algorithm : 3.5 Marks
- Testing of MAX-HEAPIFY algorithm on given example: 3.5 Marks
- Analysis of Time Complexity of HEAPIFY: 3 Marks

Answer

Max-Heapify Procedure

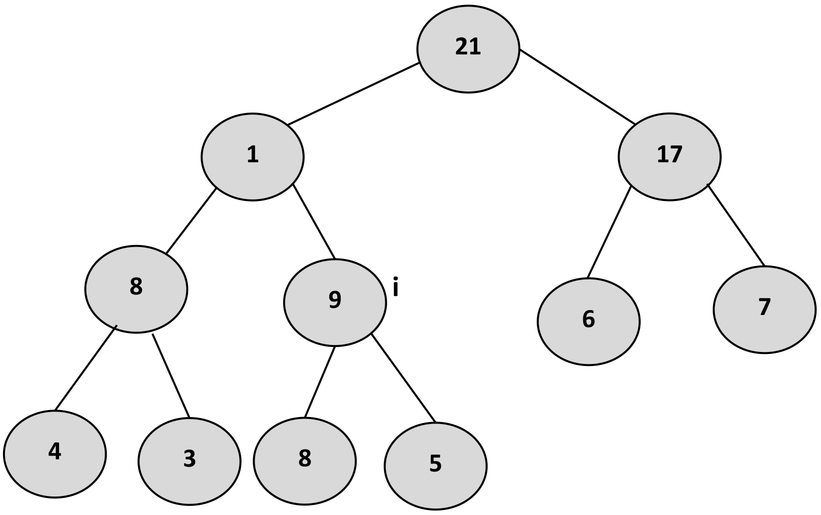
/***Max-Heapify :** Given a tree that is a heap except for node i, Max-Heapify function arranges node i and it's subtrees to satisfy the heap property.*/

```
MAX-HEAPIFY(A, n, i)
{
    l ← LEFT(i);
    r ← RIGHT(i);
    if  (l ≤ n and A[l] > A[i])
        largest = l;
    else
        largest = i;
    if  (r ≤ n and A[r] > A[largest])
        largest = r;
  if (largest != i)
  {
          A[i] ↔ largest]; // swaping
          MAX-HEAPIFY(A, n, largest);
     }
}
```

Building Max-Heap (Illustrate of operation of MAX-HEAPIFY on the array A={21, 1, 17, 8, 9, 6, 7, 4, 3, 8, 5}

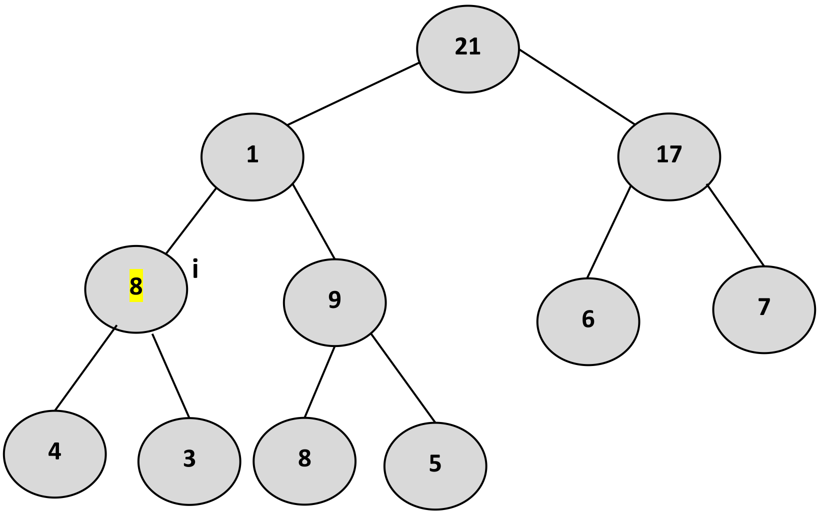
- First arrange the elements of the array into a heap shape (complete binary tree structure).
- Apply MAX-HEAPIFY(A, n, i) for i=n/2 down to 1 for the above heap shape. In this case it is starting at index i=11/2=5

Heap Shape with given array A



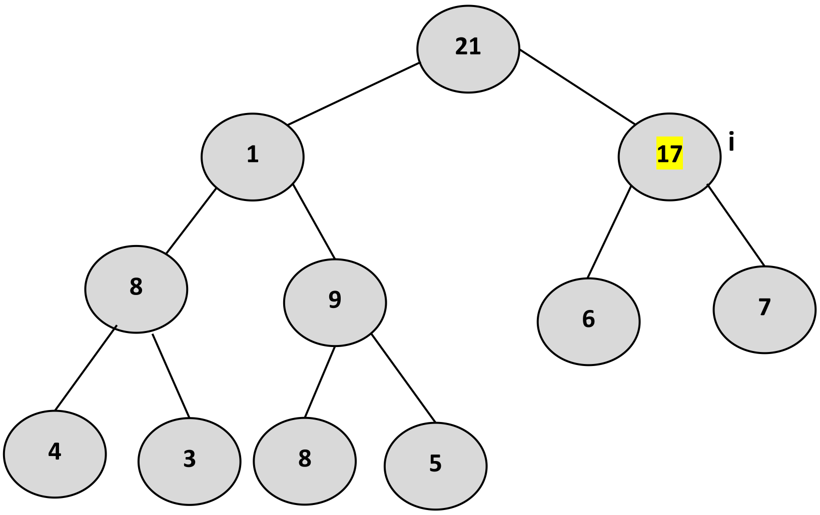
.

Heap after applying MAX-HEAPIFY(A, 11, 5) on above shape



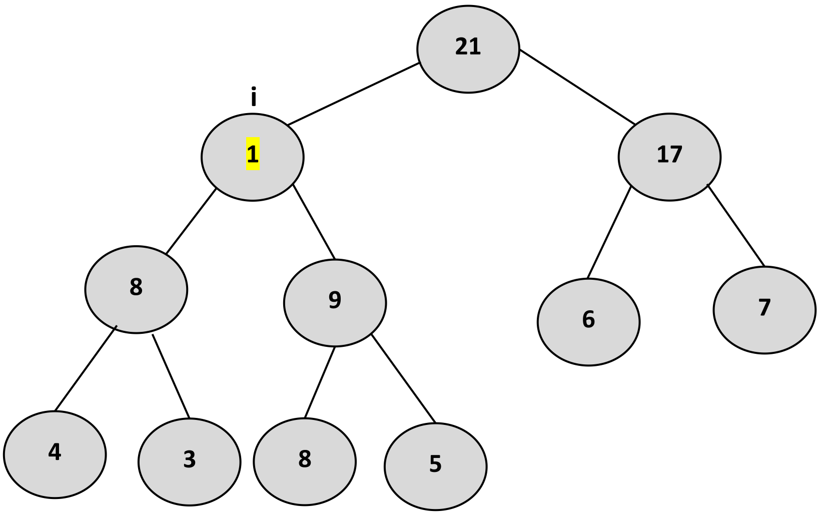
.

Heap after applying MAX-HEAPIFY(A, 11, 4) on above shape



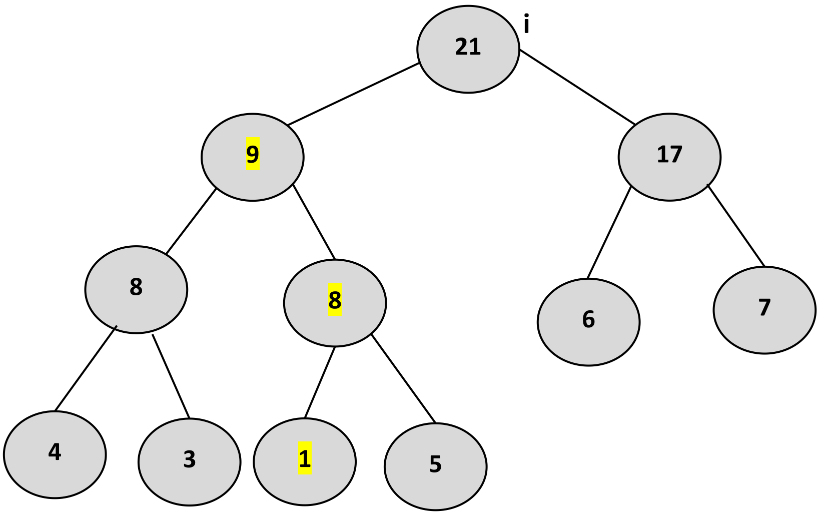
.

Heap after applying MAX-HEAPIFY(A, 11, 3) on above shape



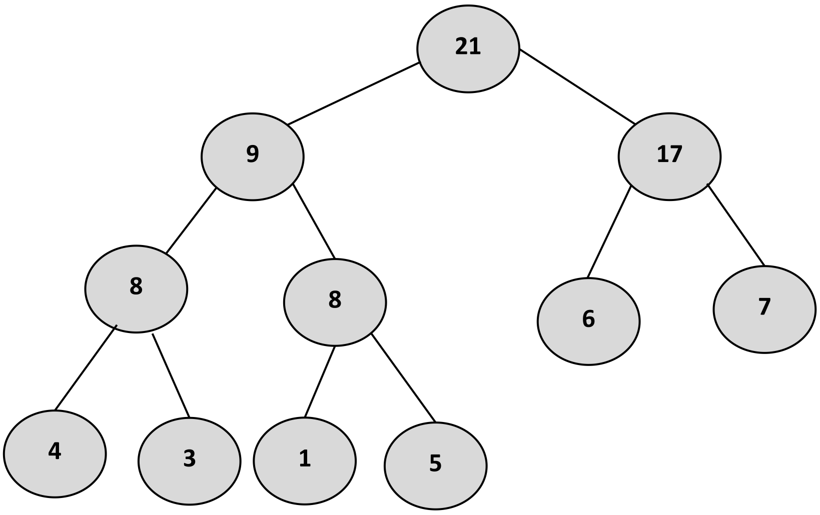
.

Heap after applying MAX-HEAPIFY(A, 11, 2) on above shape



.

Heap after applying MAX-HEAPIFY(A, 11, 1) on above shape



(This is the Max Heap)

Time Complexity of Heapify

- Time complexity for MAX-HEAPIFY is O(log n)
- Time complexity for Building a Binary Heap is O(n)

Q.No: 4 Write the PARTITION() procedure of QUICK-SORT() algorithm. Show the application of partitioning procedure at each step on the array A = { 99, 88, 77, 66, 55, 44, 33, 22, 11 }. Derive the best case time complexity of QUICK-SORT() algorithm. What is the time complexity of QUICK-SORT() on a sorted array of size 'n'?

Scheme

- PARTITION Algorithm : 3.5 Marks
- Testing of PARTITION algorithm on given example: 3.5 Marks
- Analysis of Time Complexity of QUICK-SORT: 3 Marks

Answer

PARTITION Algorithm:

```
PARTITION(A,p,r)
{
        x ← A[r]
        i ← p-1
        for j ← p to r-1
        {
             if A[j] ≤ x
             {
                     i ← i+1
                     A[i] ↔A[j]
              }
        }
         i ← i+1
         A[i] ↔A[r]
         return i
}
```

Representation of intermediate steps of pass-1

Given Data, array A = { 99, 88, 77, 66, 55, 44, 33, 22, 11 }

Pivot Element: Highlighted in yellow color (Last element is taken as Pivot)

i	j=p								r
0)	99	88	77	66	55	44	33	22	**11**
	i								j=r
	99	88	77	66	55	44	33	22	**11**
a)	**11**	99	88	77	66	55	44	33	22

Pivot Element

First Half - No element

Second Half of pass-1

Analysis of Quick Sort Algorithm

Best Case

- The best case occurs when the partition process always picks the middle element as pivot. Following is recurrence for best case.

$$T(n) = 2T(n/2) + \Theta(n)$$

- The solution of above recurrence is $\Theta(n \log n)$.

Time Complexity on Sorted Array

- If the above partition strategy is consodered, where last element is always picked as pivot, the worst case would occur when the array is already sorted in increasing or decreasing order. Following is recurrence for worst case.

$$T(n) = T(n-1) + \Theta(n)$$

- The solution of above recurrence is $\Theta(n^2)$.

Q.No:5 Write the INSERTION-SORT() algorithm and apply to the list {2, 7, 5, 1, 2}. Derive the time complexities of INSERTION-SORT() on the data that are sorted & reversely sorted respectively.

Scheme

- Correct Insertion Sort algorithm : 3.5 marks
- Bestcase & Worstcase time complexity analysis: 3.5 Marks
- Testing of Insertion Sort Algorithm on given data :3 Marks

Answer

Line No.	Insertion Sort Algorithm	Cost	Times
1	INSERTION-SORT(A)	0	
2	{	0	
3	for j←2 to length[A]	c1	n
4	{	0	
5	key←A[j]	c2	n-1
6	//Insert A[j] into the sorted sequence A[1..j-1]	0	
7	i←j-1	c3	n-1
8	while(i>0 and A[i]>key)	c4	$\sum_{j=2}^{n} \text{tj}$
9	{	0	
10	A[i+1]←A[i]	c5	$\sum_{j=2}^{n} (\text{tj} - 1)$
11	i←i-1	c6	$\sum_{j=2}^{n} (\text{tj} - 1)$
12	}	0	
13	A[i+1]←key	c7	n-1
14	}	0	
15	}	0	

To compute T(n), the running time of INSERTION-SORT on an input of n values, we sum the products of the cost and times column, obtaining

$T(n)=c1n + c2(n-1) + c3(n-1) + c4 \sum_{j=2}^{n} tj + c5 \sum_{j=2}^{n} (tj - 1) + c6 \sum_{j=2}^{n} (tj - 1) +c7(n-1)$ ………… (1)

Best case Analysis:

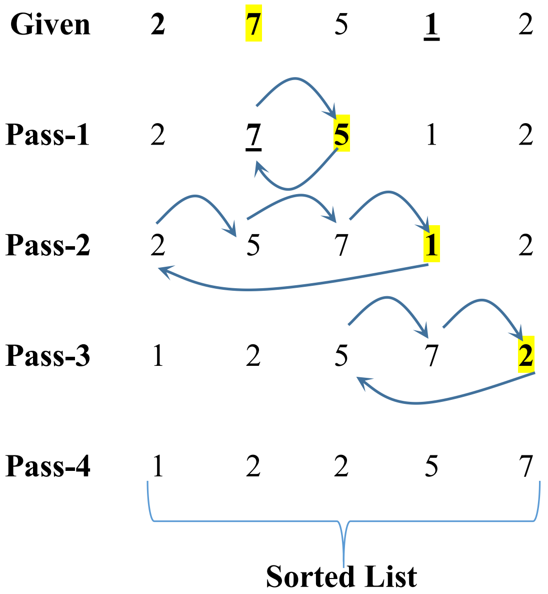
- Best case occurs if the array is already sorted.
- For each j=2 to n, we find that A[i]≤key in line number 8 when I has its initial value of j-1. Thus tj=1 for j=2 to n and the best case running time is
  $T(n)=c1n + c2(n-1) + c3(n-1) + c4(n-1) + c7(n-1) =O(n)$

Worst case Analysis:

- Worst case occurs if the array is already sorted in reverse order
- In this case, we compare each element A[j] with each element in the entire sorted subarray A[1..j-1], so tj=j for j=2 to n.
- The worst case running time is
  $T(n)=c1n + c2(n-1) + c3(n-1) + c4(n(n+1)/2-1) + c5(n(n-1)/2) + c6(n(n-1)/2) +c7(n-1)=O(n^2)$

Application of INSERTION_SORT to the list {2, 7, 5, 1, 2}

d (digit highlighted with yelooe color) - Reference element
d (digit with undeline) - Location to insert reference element



Given	**2**	**7**	5	**1**	2
Pass-1	2	**7**	**5**	1	2
Pass-2	2	5	7	**1**	2
Pass-3	1	2	5	7	**2**
Pass-4	1	2	2	5	7

Sorted List

Q.No: 6 Given an unsorted array A[1..n] where first x (x≤n) elements of the array are sorted in ascending order and rest elements of the array are sorted in descending order. Design an algorithm to sort the array in O(n) worst-case time.

Scheme

- Correct algorithm with O(n) worst-case time : 10 Marks
- Correct algorithm with other than O(n) worst-case time : 4-7 Marks
- Algorithm approaches to solution (not fully correct) : 0.5-5 Marks

Answer

```
ARRAY-MERGE(A,n, x)
{
  //Transfer first x (x≤n) elements of array A into   L array
  k=1
    for i←1 to x
    {
         L[k]←A[i]
         k←k+1
         i ←i+1
  }
  //Transfer rest n-x (x≤n) elements of array A into   R array
  k=1
  for j←n down to x+1
    {
         R[k]←A[j]
          k←k+1
          j←j-1
  }
  //Merge sorted array P ascending, Q ascending   to R ascending
  MERGE(L, x, R, n-x, A);
}
```

```
/*Merge procedure to merge ascending sorted arrays P[1..m] and Q[1..n] into array
R[1..m+n] in ascending order. */
MERGE(P, m, Q, n, R)
{
    i←1, j←1, k←1;
    while(i≤m and j≤n)
    {
         if(P[i]<B[j])
         {
              R[k]=P[i]
              k←k+1
              i←i+1
         }
         else
         {
             R[k]=Q[j]
             k←k+1
             j←j+1
         }
    }
    while(i≤m)
    {
              R[k]←P[i]
              k←k+1
              i←i+1
   }
    while(j≤n)
    {
             R[k]←P[j]
             k←k+1
             j←j+1
   }
}
```

*******THE END-ONLINE EXAM-MID-DAA-AUTUMN 2020 SOLUTION & EVALUATION SCHEME*******