Date_____ Page No._____

Tutorial-3

Ans 1
```
while (low <= high)
{
 mid = (low + high)/2
 if (arr[mid] == key)
     return true
 else if (arr[mid] > key)
         high = mid - 1
 else  low = mid + 1
}
return false
```

Ans 2 Iterative insertion sort:

```
for (int i=1; i<n; i++)
{
   j = i-1;
   x = A[i];
   while (j>-1 && A[j]>n)
   {
      A[j+1] = A[j];
      j--;
   }
   A[j+1] = n;
}
```

Date _____ Page No. _____

Recursive insertion sort:

```
void insertionsort (int arr[], int n)
{
  if (n<=1)
  return
  insertionsort (arr, n-1)
  int last = arr[n-1]
  j=n-2
  while (j>=0 && arr[j]>last)
  {
     arr[j+1] = arr[j]
     j--;
  }
  arr[j+1] = last;
}
```

Insertion sort is online sorting because whenever a new element come, insertion sort define its right place.

Date_____ Page No._____

Ans 3 Bubble sort → $O(n^2)$
Insertion → $O(n^2)$
Selection → $O(n^2)$
Merge → $O(n \log n)$
Count → $O(n)$
Quick → $O(n \log n)$
Bucket → $O(n)$

Ans 4 Online sorting → Insertion sort
Stable sorting → Merge sort, Insertion, Bubble
Inplace sorting → Bubble, Insertion, Selection

Ans 5 Iterative Binary Search:

```
while (low <= high)
{
    int mid = (low + high) / 2
    if (arr[mid] == key
        return true
    else if (arr[mid] > key)
        high = mid - 1
    else
        low = mid + 1
```


Teacher's Signature_____

Date _____ Page No. _____

Recursive:
```
while (low <= high)
{
    int mid = (low + high)/2
    if (arr[mid] == key)
        return true;
    else if (arr[mid] > key)
        binary search (arr, low, mid-1)
    else
        binary search (arr, mid+1, high
}
return false;
```

Ans 6 $T(n) = T(n/2) + T(n/2) + c$

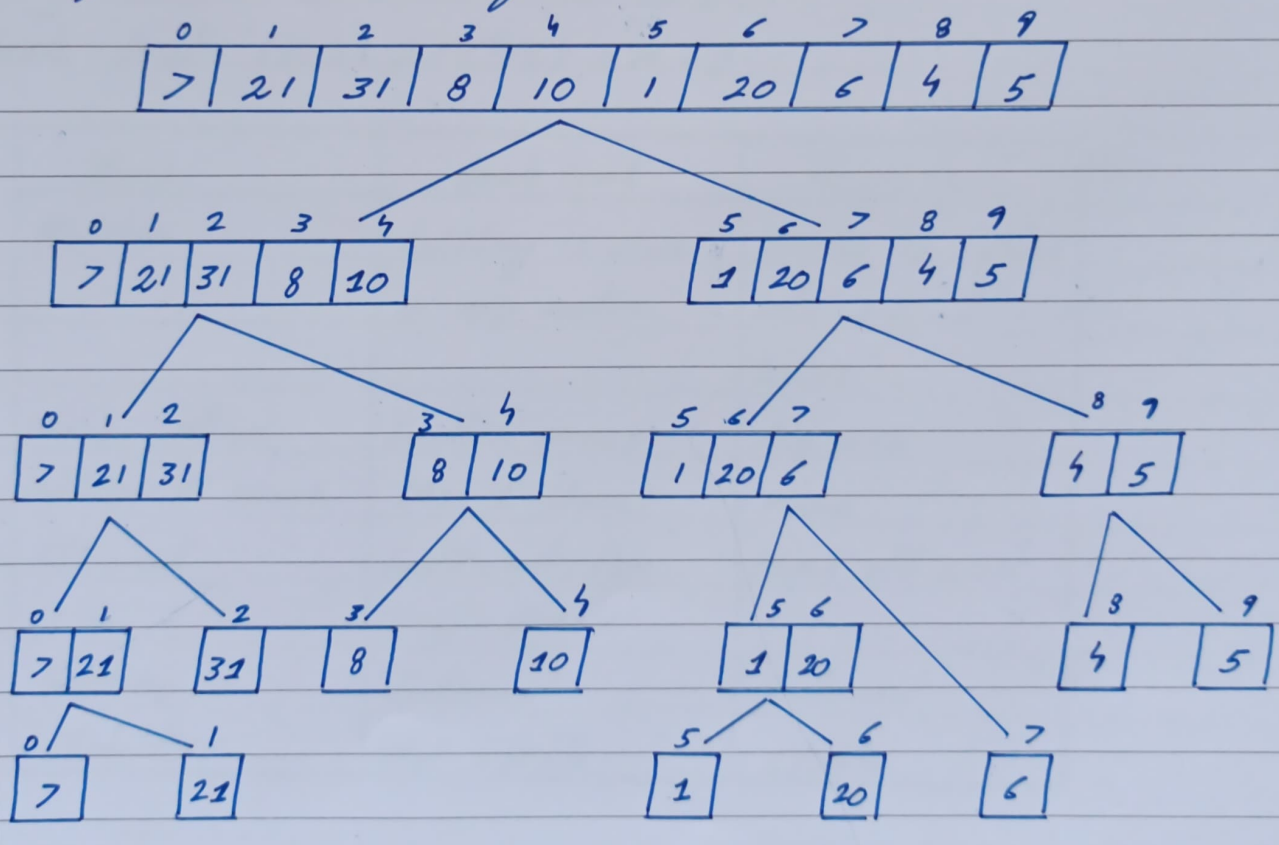
Ans 7
```
for (int i=0; i<arr.size(); i++)
{
    if (m.find (target - arr[i]) = m.end())
        m[arr[i]] = i;
    else
        cout << i << " " << mp[arr[i]];
}
```

Mohan Teacher's Signature _____

Date_____ Page No._____

Ans 8 Quicksort is the fastest general purpose sort. In most practical situation, quicksort is the method of choice.
If stability is important & space is available, merge sort might be best.

Ans 9 Inversion indicates - how far or close the array is from being sorted.



Inversions = 31

Ans 10 Worst Case: Occurs when the picked pivot is always an extreme element. It happens when input array is sorted as reverse sorted & either first or last element is picked as pivot.

Best Case: Best case occurs when pivot element is the middle element.

Ans 11 Merge Sort: $T(n) = 2T(n/2)$ to (n)

Quick Sort: $T(n) = 2T(n/2) + n + 1$

	Basis	Quick Sort	Merge Sort
•	Partition	Splitting is done in any ratio	Array is parted into just 2 halves.
•	Works well on	Smaller array	Any size
•	Addition of space	Less (inplace)	More
•	Efficient	Inefficient for larger array	More efficient
•	Sorting	Internal	External
•	Stability	Not Stable	Stable

Date

Page No.

Ans 14 We will use Merge Sort because we can divide the 4 GB data into 4 packets of 1 GB & sort them separately & combine later.

- Internal: All the data is stored in memory at all times.

- External: All the data is stored outside memory & only loaded into memory in small chunks.

Mohan

Teacher's Signature