Palak Tayal
F, 18

Tutorial -3

Que1 linear Search

```
int LinearSearch (int *arr, int n, int key)
{
    for i<-0 to n-1
        if arr[i] = Key
            return i

    return -1
```

Que2 Insertion Sort

Iterative:

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

Recursion:

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

* Insertion sort is called 'online sort because it doesn't need to know anything about what values it'll sort and info is requested while algorithm is running.

Que3 Complexity of all Sorting algos

Sorting	Best	Average	Worst
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$
Heap	$O(n.\log n)$	$O(n.\log n)$	$O(n.\log n)$
Quick	$O(n.\log n)$	$O(n^2)$	$O(n.\log n)$
Merge	$O(n.\log n)$	$O(n.\log n)$	$O(n.\log n)$

Que4 Implace Sorting | Stable Sorting | Online Sorting

Implace Sorting	Stable Sorting	Online Sorting
Bubble	Merge	Insertion
Selection	Bubble	
Insertion	Insertion	
Quick	Count	
Heap		

Que5 Binary Search

Iterative:

```
int BinarySearch(int arr[], int l , int r , int x)
{
    while (l<=r)
    {
        if (arr [m]= x)
            return m;
        if(arr[m]<n)
            l = m+1;
        else
            r=m-1;
    }
    return -1;
}
```

Recursive:

```
Bool Binary Search (int * arr, int l, int r, int key)
{ if (l>r)
      return False;
  int mid = (l+r)/2;
  if(arr[mid] == key)
      return True;
  else if( arr[mid] < key)
      return BinarySearch(arr, mid+1 , r, key);
  else
      return BinarySearch(arr, l,mid-1, key);
}
```

Ques 6

Recurrance Relation for binary search

$$T(n) = T(n/2) + 1$$

$$n = n/2$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 1$$

$$T(n) = T\left(\frac{n}{4}\right) + 1 + 1$$

$$n = n/4$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1$$

$$T(n) = T\left(\frac{n}{8}\right) + 1 + 1 + 1$$

$$T(n) = T\left(\frac{n}{8}\right) + 3$$

$$T(n) = T\left(\frac{n}{2^k}\right) + k$$

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$\log n = k$$

$$T(n) = T\left(\frac{n}{n}\right) + \log n$$

$$T(n) = 1 + \log n$$

$$\boxed{T(n) = O(\log n)}$$

Que7

Find 2 indexes such as $A[i] + A[j] = K$

```
for(i=0; i<n; i++)
{
    for (int j=0; j<n; j++)
    {
        if (a[i]+ a[j] == k)
            printf("%d",i,j);
    }
}
```

Que8 Which Sorting is best for practical uses? Explain

Quick sort is fastest general-purpose sorting. In most practical situation quick sort is the method of choice as stability is important and space is available, merge sort might be best.

Que9 Inversion:

A pair (A[i], A[j]) is said to be inversion if A[i] > A[j] i<j.

arr[] = {7, 21, 31, 8, 10, 1, 20, 6, 4, 5}

Inversions = 31

Que10 Worst case $O(n^2)$:

The worst case occurs when the pivot element is an extreme (smallest / largest) element. This happens when input array is sorted or reverse sorted and either first or last element is selected as pivot.

Best case $O(n.\log n)$:

The best case occurs when we'll select pivot element as a mean element.

Que11.

	Best	worst
Merge Sort	O(n.logn)	O(n.logn)
Quick Sort	O(n.logn)	O(n2)

In quick sort, array of element is divided into 2 parts repeatedly untill it is not possible to divide it further.
In merge sort, Elements are split into subarray (n/2) again and again untill only 1 element is left.

Que12. Stable Selection sort :

```
for(int i=0; i<n-1; i++)
{
   int min=i;
   for(int j=i+1; j<n; j++)
   { if (a[min]> a[j])
        min =j;
   }
   int key= a[min];
   while (min>i)
   {
      a[min]= a[min-j];
      min--;
   }
   a[i] = key;
}
```

Que 13 A better version of bubble sort, known as m bubble sort, includes a flag that is set of a exchange is made after an entire pass over. If no exchange is made than it should be called the array is already order because no two elements need to be switched.

```
void bubble (int arr[ ], int n)
{
  for (int i=0; i<n; i++)
  { int swaps =0;
    for(int j=0 ; j<n-1; j++)
    { if (arr[j] > arr [j+1])
      {
        int t= arr[j];
        arr[j]= arr[j+1];
        arr[j+1] = t;
        swap++;
      }
    }
    if (swap ==0)
      break;
  }
}
```