Quick sort-

```
void quicksort(int l, int u, int *A)

{
  int i, j, key, t;
  if(l<u)
  {
      i = l+1;
      j = u;
      key = A[l];
     do
     {
       while(A[i]<key && i<=j)
           i = i+1;
       while((A[j]>key) && i<=j)
           j = j-1;
       if(i<j)
       {
           t = A[i];
           A[i] = A[j];
           A[j] = t;
       }
     }while(i<j);
    t = A[l];
    A[l] = A[j];
    A[j] = t;
   quicksort(l, j-1);
   quicksort(j+1, u);
  }
}
```

Insertion sort

```
Void InsertionSort :: sort ()
{
    int v;
    for(int i=0; i<= n-1; i++)
    {
        v= ins[i];
        int j= i-1;
        while (j >=0 && ins [j] >v)
        {
            ins [j+1] = ins [j];
            j= j-1;
        }
        ins [j+1] =v;
    }
}
```

Bubble sort

```
Void BubbleSort::sort()
{
    int temp;
    for(int i=0; i<=n-2; i++)
    {
      for(int j=0; j<=n-2-i; j++)
        {
            if(a[j+1]<a[j])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
}
```

Merge sort

```
void mergsort::MSort(int low, int high, int mid)
{
        int i, s, k;
    int h=i=low, s=mid+1;
  while((h<=mid) && (s<=high))
  {
     if(a[h]<=a[s])
     {
        b[i]=a[h];
        h=h+1;
     }
     else
     {
        b[i]=a[s];
        s=s+1;
     }
     i=i+1;
  }
  if(h>mid)
  {
    for(k=s; k<=high; k++)
    {
       b[i]=a[k];
       i=i+1;
    }
  }
  else
    {
      for(k=h; k<=mid; k++)
      {
         b[i]=a[k];
         i=i+1;
      }
    }
    for(k=low; k<=high; k++)
        a[k]=b[k];
}
```

Merge sort :-

```
    merge (int left, int mid, int right)
{
    int l, k, i;
    int *b;
    l = left;
    k = 0;
    i = mid+1;
  while (l<=mid && i<=right)
  {
      if (a[l]<a[i])
          b[k++] = a[l++];
      else
          b[k++] = a[i++];
  }
  while (i<=right)
  {
      b[k++] = a[i++];
  }
  while (l<=mid)
      b[k++] = a[l++];
  for(int i = left; i<=right; i++)
  {
      a[i] = b[k++];
  }
}
void merge sort(int left, int right)
{
    int mid;
    if (left<right)
    {
        mid = (left+right)/2;
        mergesort (left, mid);
```

```
        mergesort(mid+1, right);
        merge(left, mid, right);
    }
}
```

In this merge sort we have to give array of unsorted list. At the first it can be divided into 2 parts, each 2parts can be divided into another 2 half parts uptowe get individual elements. After the division compare the adjust element and arrange into assending order. upto hole list in sorted order.

Selection sort:=

```
void selectionsort()
{
    int p;
    for(int i=length-1; i>0; i--)
    {
        p=maxposition(i);
        swap(p,i);
    }
}
int maxposition(int length)
{
    int p=0;
    int max=a[0];
    for(int i=1; i<=length; i++)
    {
        if(a[i]>max)
        {
            max=a[i];
            p=i;
        }
    }
    return p;
}
```

```
void quickSort (ele a[], int L, int R)
{
   int P, i, j;
   ele temp;
   if (L < R){
        i=L; j=R+1;
        P = a[L].key;
        do {
             do i++ while (a[i].key<P);
             do j++ while (a[j].key >P);
             if( i<j ) swap (a[i], a[j], temp);
        }
        while( i<j );
        swap (a[L], a[j], temp)
        quickSort (a[L],a[j], t
        quickSort (a, L, j-1)
    }  quickSort (a, j+1, R);
}
```

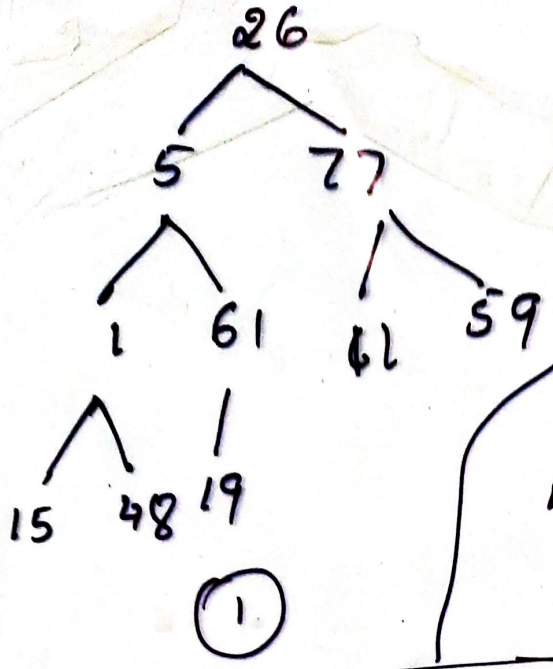
										L	R
26	5	37	1	61	11	59	15	48	19	1	10
[11	5	19	1	15]	26	[59	61	48	37]	1	5
[1	5]	11	[19	15]	26	[59	61	48	37	1	2
1	5	11	[19	15]	26	[59	61	48	37]	4	5
1	5	11	15	19	26	[59,	61	48	37]		
1	5	11	15	19	26	[48,	37]	59	[61]	7	10
1	5	11	15	19	26	37	48	59	[61]	7	8
1	5	11	15	19	26	37	48	59	61	10,	
1	5	11	15	19	26	37	48	59	61		

```
void    Swap (int i,int j)
{   int t=a[i];
    a[i]=a[j];
    a[j]=t;
}
```

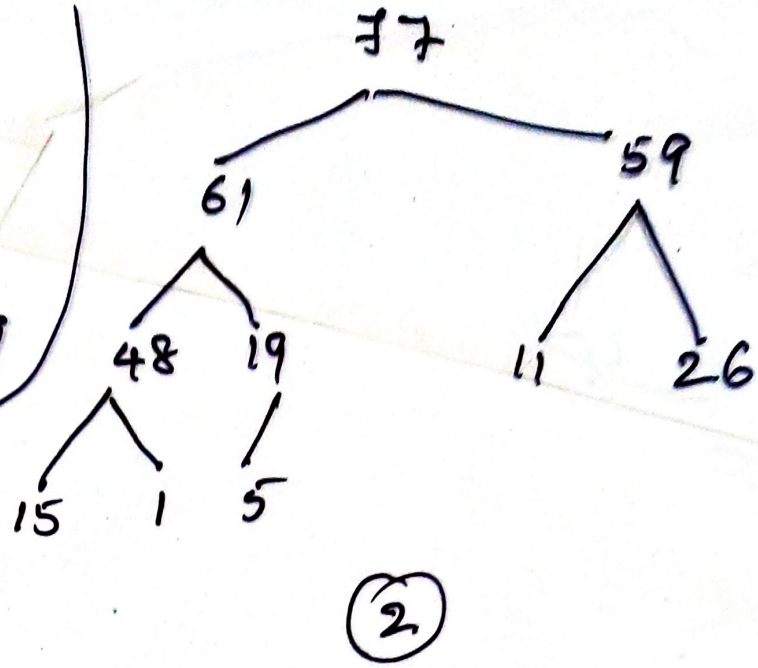
Insertion sort:

```
void    insertsort()
{
  for(int i=0; i<length; i++)
    {
      insert(i);
    }
}
void insert(int i)
{  int ele=a[i];   // ele is given the value of 1st
                      ele in array
  for(int j=i-1 ; j>=0 && ele<a[j] ; j--)
     a[j+1]=a[j]   // element inserted from an array
     a[j+1]=ele;      is sorted.
}
```

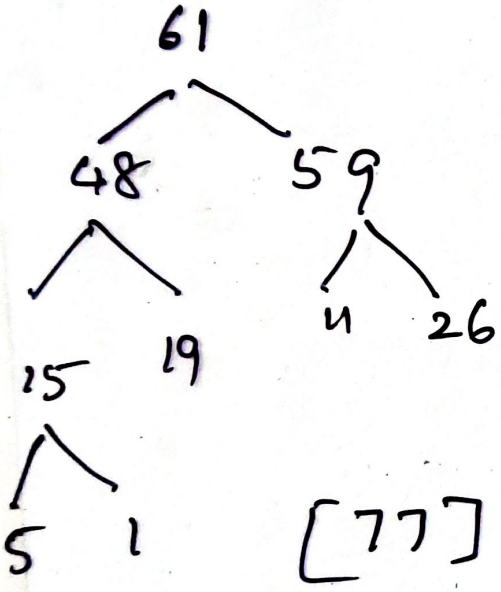
```
void    bubblesort()
{   // To peform bubblesort
    for(int i=0; i<length; i++)
    {  for(int j=0; j<length-1-i; j++)
       {   if (a[j]>a[j+1])
           // condn is checked & if true swap is occured
               swap(j, j+1);
       }  }
}
```



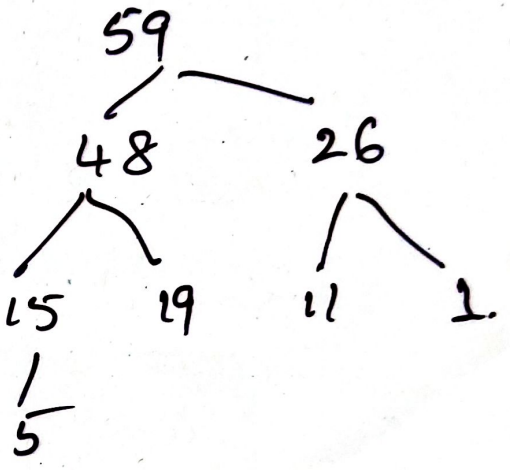
(1)



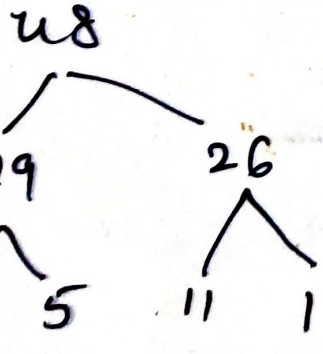
(2)



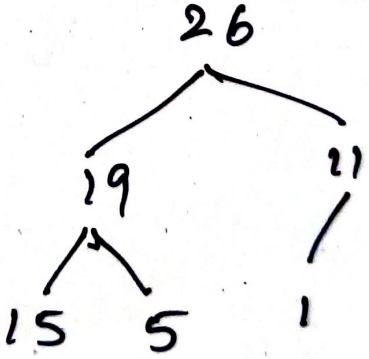
[77]



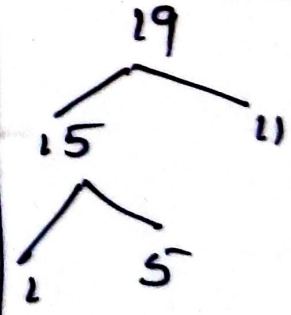
[61 77]



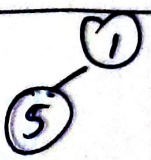
[59 61 77]

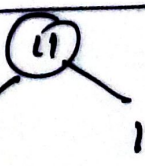


[48 59 61 77]



26 48 59 61 77





15 19 26 ....

19 26 48 59 61 77

Heap Sort —

```
void adjust(int *a, int root, in n)
{
    int e = a[root].
  for(int j = 2 * root ; j<=n ; j = 2)
  {
      if (j<n && a[j] < a[j+1]  j++
      if (e >= a[j] breaks;
        a[j/2] = a[j];
  }
  a[j/2] = e.
  }
void  HeapSort-(int a, int n)
  {
    for(i = n/2   i>=1;  i--)   //Heapify
          Adjust(a, i, n)

    for(i=n-1;  i>=1;  i--)
    {     swap(a[1], a[i+1];
          Adjust (a, 1, i);
    }

}.
```