Sorting - Arranging content in Ascending/Descending order

Bubble Sort

I	A[0]	A[1]	A[2]	A[3]	A[4]
0	45	~~90~~ 23	~~23~~ ~~90~~ 34	~~34~~ ~~90~~ 56	~~56~~ 90
1	~~45~~ 23	~~23~~ ~~45~~ 34	~~34~~ 45	56	
2	23	34	45		
3	23	34			

C++ Function to arrange elements of an array in Ascending order using Bubble Sort

```
void BubbleA(int A[],int n)
{
 for (int I=0;I<N-1;I++)
 {
   for (int J=0;J<N-I-1;J++)
     if (A[J]>A[J+1])
     {
       int T=A[J];
       A[J]=A[J+1];
       A[J+1]=T;
     }
}
```

C++ Function to arrange elements of an array in Descending order using Bubble Sort

```
void BubbleD(int A[],int n)
{
  for (int I=0;I<N-1;I++)
  {
    for (int J=0;J<N-I-1;J++)
      if (A[J]<A[J+1])
      {
        int T=A[J];
        A[J]=A[J+1];
        A[J+1]=T;
      }
}
```

Selection Sort

I	A[0]	A[1]	A[2]	A[3]	A[4]
0	45 Small=~~0~~ 4 So,we will swap Positions 0 and 4	90	55	63	28
1	28	90 Small=~~1~~ ~~2~~ 4 So,we will swap Positions 1 and 4	55	63	45
2		45	55 Small=2 No change in Small So,no swapping of positions needed	63	90
3			55	63 Small=3 No change in Small So,no swapping of positions needed	90

C++ Function to arrange elements of an array in

Ascending order using Selection Sort

```
void SelectionA(int A[],int n)
{
  for (int I=0;I<N-1;I++)
  {
    int Small=I;
    for (int J=I+1;J<N;J++)
      if (A[Small]>A[J])
        Small=J;
    if (Small!=I)
    {
      int T=A[Small];
      A[Small]=A[I];
      A[I]=T;
    }
}
```

C++ Function to arrange elements of an array in

Descending order using Selection Sort

```
void SelectionD(int A[],int n)
{
  for (int I=0;I<N-1;I++)
  {
    int Big=I;
    for (int J=I+1;J<N;J++)
      if (A[Big]<A[J])
        Big=J;
    if (Big!=I)
    {
      int T=A[Big];
      A[Big]=A[I];
      A[I]=T;
    }
}
```

Insertion Sort

I	A[0]	A[1]	A[2]	A[3]	A[4]
	95	90	55	63	28
1	90	~~90~~ 95 Temp=A[1]=90			
2	~~90~~ 55	~~95~~ 90	~~55~~ 95 Temp=A[2]=55		
3	55	~~90~~ 63	~~95~~ 90	~~63~~ 95 Temp=A[3]=63	
4	~~55~~ 28	~~63~~ 55	~~90~~ 63	~~95~~ 90	~~28~~ 95 Temp=A[4]=28

C++ Function to arrange elements of an array in

Ascending order using Insertion Sort

```
void InsertionA(int A[],int n)
{
  for (int I=1;I<N;I++)
  {
    int Temp=A[I],J=I-1;
    while(Temp<A[J])
    {
      A[J+1]=A[J];
      J--;
    }
    A[J+1]=Temp;
  }
}
```

C++ Function to arrange elements of an array in

Descending order using Insertion Sort

```
void InsertionD(int A[],int n)
{
  for (int I=1;I<N;I++)
  {
    int Temp=A[I],J=I-1;
    while(Temp>A[J])
    {
      A[J+1]=A[J];
      J--;
    }
    A[J+1]=Temp;
  }
}
```

Binary Search

Prerequisite - The array content should be sorted (Ascending/Descending)

Let us assume, an array A[10] is arranged in ascending order. The following are the steps to search for a value 85 from the array using Binary Search.

	A[10]	Step	Data to searched=85
0	23	1	LB=0;UB=9;MID=(0+9)/2=4;As (A[4]<85) LB=MID+1=5
1	45	2	LB=5;UB=9;MID=(5+9)/2=7;As (A[7]>85) UB=MID-1=6
2	67	3	LB=5;UB=6;MID=(5+6)/2=5;As (A[5]<85) LB=MID+1=6
3	69	4	LB=6;UB=6;MID=(6+6)/2=6;As (A[6]==85) Data Found
4	73		
5	81		
6	85		
7	91		
8	95		
9	99		

Let us assume, an array A[10] is arranged in ascending order. The following are the steps to search for a value 79 from the array using Binary Search.

	A[10]	Step	Data to searched=79
0	23	1	LB=0;UB=9;MID=(0+9)/2=4;As (A[4]<79) LB=MID+1=5
1	45	2	LB=5;UB=9;MID=(5+9)/2=7;As (A[7]>79) UB=MID-1=6
2	67	3	LB=5;UB=6;MID=(5+6)/2=5;As (A[5]>79) UB=MID-1=4
3	69	4	LB=5;UB=4; As LB>UB, Data Not Found
4	73		
5	81		
6	85		
7	91		
8	95		
9	99		

C++ Function to SEARCH for a value in an array arranged in
Ascending order using Binary Search

```
int BinSearch(int A[],int N,int Data)
{
  int LB=0,UB=N-1,MID,Found=0;
  while (LB<=UB && !Found)
  {
    MID=(LB+UB)/2;
    if (A[MID]>Data)
      UB=MID-1;
    else if (A[MID]<Data)
      LB=MID+1;
    else
      Found++;
  }
  return Found;
}
```

C++ Function to SEARCH for a value in an array arranged in
Descending order using Binary Search

```
int BinSearch(int A[],int N,int Data)
{
  int LB=0,UB=N-1,MID,Found=0;
  while (LB<=UB && !Found)
  {
    MID=(LB+UB)/2;
    if (A[MID]>Data)
      LB=MID+1;
    else if (A[MID]<Data)
      UB=MID-1;
    else
      Found++;
  }
  return Found;
}
```

Merging

C++ Function to
MERGE two ascending order array

```
void Merge(int A[],int B[],int C[],
           int N,int M,int &L)
{
  int I=0,J=0;
  L=0;
  while (I<N && J<M)
    if (A[I]<B[J])
      C[L++]=A[I++];
    else if (A[I]>B[J])
      C[L++]=B[J++];
    else
    {
      C[L++]=A[I++];
      J++;
    }
  while(~~I<N~~ J<M)
    C[L++]=A[I++];
  while(I<N)
    C[L++]=B[J++];
}
```

Input Array A	Input Array B	Output Array C
A[0]=14	B[0]=13	C[0]=13
A[1]=23	B[1]=23	C[1]=14
A[2]=56	B[2]=59	C[2]=23
	B[3]=65	C[3]=56
	B[4]=78	C[4]=59
		C[5]=65
		C[6]=78

Note: If we require to merge an Ascending order Array A and a Descending Order Array B to produce an Ascending order Array C.

Initialization of Merge function will be changed to [`int I=0,J=M-1;`]

Conditions will be modified to [`I<N && J>=0`]

Change in subscript of A will be same as `I++` but B will be `J--`