SORTING

Sorting means arranging the data in a particular order. Strings can be arranged in alphabetical order and numeric can be arranged in ascending or descending order. Here, we will use two logics for sorting

Bubble sort: For ascending order

Here, two adjacent numbers are compared and if 2nd is smaller than first then, their position is swapped or interchanged. This method works for small data, however, for large data, the number of iterations (cycles) are very large and the result may take very long time.

19	**29**	**3**	**46**	**5**

Compare 1st and second numbers, they are already in ascending order. Now compare, 2nd and 3rd numbers and swap their positions.

19	**3**	**29**	**46**	**5**

Now compare 3rd and 4th numbers, there is no change so we compare 4th and 5th and swap

19	**3**	**29**	**5**	46

This is the order after 1st cycle, the last position is now occupied by largest number. Now begin the next cycle and again compare 1st and 2nd numbers and swap

3	**19**	29	5	46

Now compare, 2nd and 3rd numbers, no change. Now compare 3rd and 4th numbers and swap

3	**19**	**5**	29	46

Now compare 4th and 5th , they are already sorted.

Start 3rd cycle and compare 1st and 2nd numbers, there is no change, then compare 2nd and 3rd and swap.

3	**5**	**19**	29	46

Now compare 3rd and 4th, they are already sorted. Then start the next cycle, and compare 1st and 2nd numbers, there is no change.

Second method for ascending order. It is another method for sorting small data.

We take the same array and an empty box named T, Assume that the first number is smallest and keep it in the empty box from 1st position.

19	**29**	**3**	**46**	**5**	T

Compare 19 with the 2nd number, if it is smaller, there is no change otherwise interchange the position and continue comparing with next numbers. See how it is done

	29	3	46	5	T
					19

Compare 29 and 19 no change, compare 3 and 19, Swap , now we have

	29	19	46	5	T
					3

Compare 46 and 3 no change and compare 5 and 3 no change, so now put 3 in first box and 2nd number in T.

3		19	46	5	T
					29

Now compare, 29 with 3rd number 19 and swap their positions.

3		29	46	5	T
					19

Then compare 19 with 4th number 46, no change, then with 5th number 5 and swap positions.

3		29	46	19	T
					5

Since 5 is the smallest of the remaining numbers, put it in box 2.

3	**5**	29	46	19	T

Put 3rd number 29 in T and continue comparing with 4th and 5th numbers. Continue in this manner until all the numbers are sorted.

Swap statement

This statement is used to interchange the values of two variables. The syntax is,
SWAP A,B
Using this statement, the value of A is assigned to variable B and the value of B is assigned to the variable A

Problem: Write a program in BASIC to arrange the names in alphabetical order using SWAP command.
Program: Bubble sort

```
10  CLS
20 REM ***ALPHABETICAL ORDER BUBBLE SORT 1ST METHOD***
30 INPUT "No.of students"; A
40 DIM N$(A)
50 FOR I = 1 TO A
60 INPUT "Name of student"; N$(I)
70 NEXT I
75 PRINT "NAME IN ASCENDING ORDER"
80 X = A - 1
90 FOR I = 1 TO A
100 FOR J = 1 TO X
110 IF N$(J) < N$(J + 1) THEN 130 ELSE 120
120 SWAP N$(J), N$(J + 1)
130 NEXT J
140 X = X - 1
150 NEXT I
160 FOR I = 1 TO A
170 PRINT N$(I),
180 NEXT I
190 END
```

* In this method after 1st cycle, the name starting with letter having highest ASCII code is placed at the last position. After 2nd cycle, the 2nd last position is filled and so on. I is also counting the number of cycle being performed. So after each cycle we are reducing the maximum value of J

Method 2

```
10  CLS
20 REM ***ALPHABETICAL ORDER BUBBLE SORT 1ST METHOD***
30 INPUT "No.of students"; A
40 DIM N$(A)
50 FOR I = 1 TO A
60 INPUT "Name of student"; N$(I)
70 NEXT I
75 PRINT "NAME IN ASCENDING ORDER"
80 X = 1
90 FOR I = 1 TO A
100 FOR J = A TO X STEP -1
110 IF N$(J) < N$(J - 1) THEN 120 ELSE 130
120 SWAP N$(J), N$(J - 1)
130 NEXT J
140 X = X + 1
150 PRINT N$(I),
160 NEXT I
170 END
```

Method 3

```
10 CLS
20 REM*ALPHABETICAL ORDER*
30 INPUT "NO. OF STUDENTS"; A
40 DIM N$(A)      This reserves A 1-D array slots for variable N$
50 FOR I = 1 TO A
60 INPUT "NAME "; N$(I)  {here you have to enter the name, FOR ……NEXT loop
                         is used so it will ask you to enter A number of
                         names }
70 NEXT I
80 FOR I = 1 TO A
90 FOR J = 1 TO A
100 IF N$(I) > N$(J) THEN 120
110 SWAP N$(I), N$(J)
120 NEXT J
130 NEXT I
140 PRINT "NAMES IN ASCENDING ORDER"
150 FOR I = 1 TO A
160 PRINT N$(I),
170 NEXT I
180 END
```

Another algorithm:

Logic

1	2	3	4	5
Guava	Mango	Apple	Orange	Grapes

Cycle 1: Compare 1 and 2, G comes before M so no change. ASCII code of G is less than ASCII code of M. Then compare 1 with 3, A comes before G so they SWAP their positions and order becomes

1	2	3	4	5
Apple	Mango	Guava	Orange	Grapes

Now 1 is compared with 4, no change and then 1 is compared with 5 and there is no change. So number 1 is fixed.

Cycle 2: compare 2 with 3, G comes before M so they swap positions, order becomes

1	2	3	4	5
Apple	Guava	Mango	Orange	Grapes

Now the new 2 is compared with 4, there is no change, then with 5, they SWAP positions. This is because after comparing G of Guava and Grapes, next compare u and r. Since r comes before u Grapes has smaller ASCII value. The 2nd position is also now fixed. The new order is

1	2	3	4	5
Apple	Grapes	Mango	Orange	Guava

Cycle 3: Compare 3 with 4, no change, then with 5 , SWAP . Continue in this manner till all comparisons are made.

1	2	3	4	5
Apple	Grapes	Guava	Orange	Mango

Cycle 4

1	2	3	4	5
Apple	Grapes	Guava	Mango	Orange

Program:

```
10 CLS
20 REM*ALPHABETICAL ORDER*
30 INPUT "NO. OF STUDENTS"; A
40 DIM N$(A)      This reserves A 1-D array slots for variable N$
50 FOR I = 1 TO A
60 INPUT "NAME "; N$(I) {here you have to enter the name, FOR .....NEXT loop
                        is used so it will ask you to enter A number of
                        names }
70 NEXT I
75 PRINT "NAMES IN ASCENDING ORDER"
80 FOR I = 1 TO A       {Now the sorting loop has started. We need 2
                     counters, one for first name and 2nd for the name to
                     be compared.
90 FOR J = I+1 TO A     1st name is compared with 2nd, 3rd so on not with
                        itself
100 IF N$(I) < N$(J) THEN 120  ASCII codes of 1st letters of two names
                   labeled as N$(I) &N$(J)are compared. remember ASCII code
                   of A is smallest and it increases till Z.
110 SWAP N$(I), N$(J)   N$(I) and N$(J) change positions
120 NEXT J
130 PRINT N$(I),        Once N$(1) is compared with all other names, the
                   first position is fixed, so we can print it. Comma is
                   used to print in same row.
140 NEXT I
150 END
```

Problem: Write a program in BASIC to arrange the numbers in ascending order without using SWAP command.
Here we are using READ… DATA instead of INPUT command.

Program:

```
10 CLS
```

```
20 INPUT n          Total numbers to be arranged
25 dim a(n)         This reserves n 1-D array slots for variable a(i)
30 FOR i = 1 TO n
40 READ a(i)
50 NEXT i
55 PRINT " Nos. in Ascending order"
60 FOR i = 1 TO n              we compare number at position i with
70 FOR j = i + 1 TO n                number at position j
80 IF a(i) < a(j) THEN 120     for first cycle a(i) is a(1) so we compare
90 t = a(j)         a(1) with next number, if it is smaller their positions
100 a(j) = a(i)   are interchanged and we continue comparing the new a(1)
110 a(i) = t        with the remaining numbers
120 NEXT j
130 PRINT a(i)     after first cycle of changing j, a(1) is the smallest
                   number so it is printed
140 NEXT I          then we move to a(2) and compare it with a(3) and so on
150 DATA 1000,9,0,250,769
160 END
a(1) = 1000 a(2)=9 a(3)=0 a(4)=250  a(5)=769
T =a(2)=9
a(2)=a(1)=1000
a(1)=T=9
a(1)with a(3)
T=a(3)=0    a(3)=a(1)=9       a(1)=T=0
a(1) with a(4)
a(1) with a(5)

a(2) with a(3)
```

T=a(3) =9 a(3)= a(2) =1000 a(2)=T =9

a(2) with a(4)