Unit 1 - Activity 7 - Bubble Sorting Reading

First of all you need to store your data in array then you can use given code snippet for arrange date in ascending/Descending order in GW Basic

```
10 REM ASCENDING ORDER
20 CLS
30 PRINT TAB(15); "DISPLAY DATA IN ASCENDING ORDER"
40 FOR A=1 TO n STEP 5 // n= last number of your data
50 PRINT TAB(22);A
60 NEXT A
70 END
```

Similarly,

```
10 REM DESCENDING ORDER
20 CLS
30 PRINT TAB(15); "DISPLAY DATA IN DESCENDING ORDER"
40 FOR A=n TO 1 STEP -5// // n= last number of your data
50 PRINT TAB(22);A
60 NEXT A
70 END
```

```
'Bubble Algorithm Example Program

CLS
DIM numbers(1 to 10) AS INTEGER
'Declare an array with 10 places numbered
'1 to 10
DIM trigger AS INTEGER

PRINT "Enter 10 numbers to sort.."
FOR i = 0 TO 9
INPUT numbers(i)
NEXT i

CLS
FOR i = 0 TO 9
PRINT numbers(i)
'This is simply to show the original
'order of the numbers
NEXT i

PRINT "Sorting..."
DO
trigger = 0
FOR i = 0 TO 8
IF numbers(i) > numbers(i + 1) THEN
SWAP numbers(i), numbers(i + 1)
trigger = 1
END IF
NEXT i
LOOP WHILE trigger = 1


FOR i = 0 TO 9
PRINT numbers(i)
NEXT i
' This displays the sorted numbers...
```

END

Here is a routine for alpha, (or numeric), sorting of an array....

```
10 FOR Q=1 TO 9
20 IF A$(Q)=<A$(Q+1) THEN 40
30 A1$=A$(Q):A2$=A$(Q+1):A$(Q)=A2$:A$(Q+1)=A1$:X=X+1
40 NEXT:IF X>0 THEN X=0:GOTO 10
```

The example sorts an array of ten into alphabetic order. Note the for-next loop must be for one less than the number you wish to sort. I don't know how it will go with 4000, but I have used it for several hundred and it works fine. You can refine it for use with numerics, or to reverse order the listing or to look at just some of the characters in each item in the array, but the above is the usual form I devised and use.

```
3000 'sort ELEMENTS of an array, ascending (shell-sort)
3010 ' call: A(n)= array, F= 1st position, L= last position
3020 ' exit: A(n)= sorted, ascending, positions F thru L
3030 ' temp: E= Exit, H= Half, I= Incr, J= Juggle
3040 H=(L-F)/2
3050 WHILE H
3060 FOR I=F TO H+F:E=1
3070 WHILE E:E=0
3080 FOR J=I TO L-H STEP H
3090 IF A(J)>A(J+H) THEN SWAP A(J),A(J+H):E=1
3100 NEXT
3110 WEND
3120 NEXT:H=H\2
3130 WEND

4000 'sort ELEMENTS of an array, descending (bubble-sort)
4010 ' call: A(n)= array, F= 1st position L= last position
4020 ' exit: A(n)= sorted, descending, positions F thru L
4030 ' temp: E= Exit, I= Incr
4040 FOR E=-1 TO 0
4050 FOR I=F TO L-1
```

```
4060 IF A(I+1)>A(I) THEN SWAP A(I),A(I+1):L=I
4070 NEXT
4080 E=L<I
4090 NEXT
```

Below is the TB program listing of a bubble sort example program, from Brian D Hahn's 1988 book "True BASIC by Problem Solving". It creates 100 random integers, then sorts them in ascending order. Regards ... Tom M

```
! BUBBLE SORT

DIM X(1000) ! NOT MORE THAN 1000 NUMBERS
RANDOMIZE
LET N=100
FOR I=1 TO N ! GENERATE RANDOM INTEGERS IN THE ..
LET X(I)=INT(N*RND)+1 ! .. RANGE 1 TO N TO TEST
NEXT I
FOR K=1 TO N-1 ! N-1 PASSES
FOR J=1 TO N-K ! N-K COMPARISONS ON EACH PASS
IF X(J)>X(J+1) THEN
LET TEMP=X(J)
LET X(J)=X(J+1)
LET X(J+1)=TEMP
END IF
NEXT J
NEXT K
FOR I=1 TO N ! PRINT THEM OUT IN ASCENDING ORDER
PRINT USING "####":X(I);
NEXT I
END
```

To illustrate the nature of True BASIC, we first give a program that multiplies two numbers and prints the result:

```
PROGRAM product
! taken from Chapter 2 of Gould & Tobochnik
LET m = 2                          ! mass in kilograms
LET a = 4                          ! acceleration in mks units
LET force = m*a                    ! force in Newtons
PRINT force
END
```

We next introduce syntax that allows us to enter the desired values of m and a from the keyboard.

```
PROGRAM product2
INPUT m
INPUT prompt "acceleration a (mks units) = ": a
LET force = m*a                          ! force in Newton's
PRINT "force (in Newtons) ="; force
END
```

List of all primes between two selected numbers

```
10 'List primes between two selected numbers
20 CLEAR :CLS :KEY OFF
30 PRINT "List primes between two selected numbers." :PRINT
40 INPUT "Input smaller number ";SMALL
50 INPUT "Input larger  number ";LARGE
60 IF LARGE < SMALL THEN SWAP LARGE,SMALL   'Swap if entered in wrong order
70 SMALL=INT(SMALL/2)*2+1                   'Ensure small number is odd
80 LARGE=INT(LARGE/2)*2-1                   'Ensure large number is odd
90 FOR K=SMALL TO LARGE STEP 2
100   X=SQR(K)                              'Only need to check to square
root
110   F=0                                   'Count factors as a logic step
120   FOR J=3 TO X STEP 2
130     IF K/J-INT(K/J)=0 THEN LET F=F+1    'Totals up factors. F=0 is prime
140   NEXT J
150   IF F=0 THEN PRINT USING "########";K; 'Prime has no factors
160 NEXT K
170 END
```