A SET OF PSEUDOCODE PROBLEMS (rev 9.2000)
SOLUTIONS

The pseudocode that you use may depend on your course: for example. some people use **while/endwhile**, and some use **while/wend**. Some use = and <> to mean equal and not equal, whereas some use = = and **!=** - or you could write it in English**.** Some use **input** and **output**, some use **read/print** , some use **accept/display**. It doesn't really matter - pseudocode is more concerned with the logic - the control flow issues.
Write pseudocode to ...

1.1 Input the dimensions of a rectangle and print its area.

```
read length, breadth
area = length * breadth
display area
```

1.2 Input the dimensions of a rectangle and print area and perimeter.

```
read length, breadth
area = length * breadth
perimeter = 2*(length+breadth)
display area, perimeter
```

1.3 Input the length of the side of a square, and print its area. Produce an error message if the length is negative. (i.e.validation)

```
read side
if side is negative
   print error message
else
   print side*side
endif
```

1.4 Input 3 numbers and print either an "all 3 equal" or a "not all equal" message.

```
input a,b,c
if (a=b) and (a=c)
   display "all 3 equal"
else
   display "not all equal"
endif
```

1.5 Input 3 numbers, and print the biggest.

```
input a,b,c
if a>b
   bigab=a
else
   bigab=b
```

```
endif

if c>bigab
   display c
else
   display bigab
endif
```

2.1 Print the numbers 1 to 100 inclusive- with a while loop.

```
n=1
while n<=100
   display n
   n=n+1
endwhile
```

2.2 Print the numbers 1 to 100 inclusive with a for loop.

```
for n=1 to 100
   display n
endfor
```

2.3 Input 2 values into the variables: start and finish,
then print the integers from start to finish inclusive. (use for)

```
input start, finish
for n=start to finish
   print n
endfor
```

2.4 Input 2 values into the variables: start and finish,
then print the integers from start to finish inclusive. (use for)
BUT, if start is bigger than finish, don't print the numbers: an error message instead!

```
input start, finish
if start > finish
   print error message
else
   for n=start to finish
      print n
   endfor
endif
```

2.5 Input 2 values into the variables: start and finish, then print the integers from start to finish inclusive. (use for)

BUT if start is bigger than finish, swap over their values so they are now in the proper order, then print the integers as specified.

```
input start, finish
```

```
if start> finish
   interchange
endif
for n=start to finish
   print n
endfor

proc interchange
temp=start
start=finish
finish=temp
endproc
```

2.6 Input 10 numbers, and print their total.

```
sum=0
for n=1 to 10
   input number
   sum=sum+number
endfor
display sum
```

2.7 Input 10 numbers, and print their total - but any numbers>100 should be ignored, i.e should not be summed.

```
sum=0
for n=1 to 10
   input number
   if number<=100
      sum=sum+number
   endif
endfor
display sum
```

2.8 Input a count, which specifies how many numbers will follow it. Print the total of the following numbers.

(eg the input might be: 4 33 52 67 83 - where the count is 4)

```
sum=0
input count
for n=1 to count
   input number
   sum=sum+number
endfor
display sum
```

2.9 Input a series of positive (>=0) numbers, ended by a negative one. Add up the numbers, and print the total. The negative one is not to form part of the sum.

```
sum=0
input number
while number>=0
   sum=sum+number
   input number
endwhile
display sum
```

2.10 Input 100 positive (>=0) numbers. Add up the numbers, and print the total. If a negative number is encountered, the program should terminate, and print the sum so far.

```
sum=0
count=1
input number
while (count<=100)and(number>=0)
   sum=sum+number
   count=count+1
   input number
endwhile
display sum
```

2.11 Using nested fors, print an hours and minutes table, of the form:

```
0    0
0    1
0    2
0    3
...
0    59
1    0
1    1
1    2
...
```

as far as 11 hours 59 mins.

```
for hours=0 to 11
   for mins=0 to 59
      dis[play hours, mins
   endfor
endfor
```

Assume the following file pseudocode:

open "fred" for input (reading)

open "june" for output (writing)

read data items from "fred" (you may read the whole line as a string, or read it into separate variables - it depends on the problem)

write data items to "june"

close "fred"

test for end of file, eg:
while not end of file
etc

3.1 Display each line of file "fred" on the screen.

```
open "fred" for input
read line from "fred"
while not end of file
   display line
   read line from "fred"
endwhile
close files
```

3.2 Read every line in "fred", and write them to file "june"

```
open "fred" for input
open "june" for output
read line from "fred"
while not end of file
   write line to "june"
   read line from "fred"
endwhile
close files
```

3.3 Count the number of lines in "fred"

```
open "fred" for input
count=1
read line from "fred"
while not end of file
   read line from "fred"
   count=count+1
endwhile
display count
close files
```

In the following, assume that each line (record) contains a persons name and age

3.4 Display the ages of everyone called "smith"

```
open "fred" for input
read name, age from "fred"
while not end of file
   if name is "smith"
      display age
   endif
   read name, age from "fred"
endwhile
close files
```

3.5 Display the names of everyone who is over 40.

```
open "fred" for input
read name, age from "fred"
while not end of file
   if age > 40
      display name
   endif
   read name, age from "fred"
endwhile
close files
```

3.6 write the names of everyone who is over 40 onto a new file.

```
open "fred" for input
open "june" for output
read name, age from "fred"
while not end of file
   if age > 40
      write name to "june"
   endif
   read name, age from "fred"
endwhile
close files
```

3.7 Write the names of everyone who is over 40 to one file, and the names of everyone <= 40 to another file.

```
open "fred" for input
open "old" and "young" for output
read name, age from "fred"
while not end of file
   if age > 40
      write name to "old"
   else
      write name to "young"
   endif
   read name, age from "fred"
endwhile
close files
```

The following assume 2 integer arrays, a and b, with elements a[1] to a[100],and b[1] to b[100]
Note that in e.g C++, arrays are numbered from 0, - but that is a minor problem. Imagine that we choose to avoid using element number 0.

4.1 Set every element of b to 0

```
for n=1 to 100
   b[n]=0
endfor
```

4.2 Set a[1] to 1, a[2] to 4, a[3] to 9 etc

```
for n=1 to 100
   a[n]=n*n
endfor
```

4.3 Read in 100 numbers, and store them in a.

```
for n=1 to 100
   input number
   a[n]=number
endfor
```

4.4 Read in a count, then read in the data values, storing them in a.
eg- the data could be: 4, 55,1232,786,456
If the count is above 100, the program should terminate without reading any numbers.

```
input count
if count<=100
   for n=1 to count
      input number
      a[n]=number
   endfor
endif
```

4.5 Read a series of positive numbers into a. The numbers are ended by a negative one, which is not to be stored. (Assume there is not more than 100 numbers)

```
input number
n=1
while number>=0
   a[n]=number
   input number
endwhile
```

4.6 Assume that 100 numbers have already been stored in a.

Copy each one from the array a into the array b.

```
for n=1 to 100
   b[n]=a[n]
endfor
```

4.7 Assume that 100 numbers have already been stored in a.
Find the biggest value in a.

```
big=a[1]
for n=2 to 100
   if a[n]>big
      big=a[n]
   endif
endfor
```

4.8 Assume that 100 numbers have already been stored in a.
Find the position of the biggest value.

```
bigposition=1
for n=2 to 100
   if a[n]>a[bigposition]
      bigposition=n
   endif
endfor
```

4.9 Assume that 100 numbers have already been stored in a.
Search the array a for the value 9876. Either print its position, or print a 'not found' message.

```
found=no
n=1
while(n<=100) and (found = no)
   if a[n]=9876
      found=yes
   else
      n=n+1
   endif
endwhile
if found=yes
   print n
else
   print "not found"
endif
```

4.10 Read 100 numbers into a, then print the sequence in reverse order.
(eg- 123 43 -23 47 ... 667 is printed as:
667 ... 47 -23 43 123)

```
for n=1 to 100
```

```
   input a[n]
endfor

n=100
while n>=1 do        (or you may know the "FOR..DOWNTO")
   print a[n]
   n=n-1
endwhile
```

4.11 Assume that 100 numbers have already been stored in a.

Input 2 numbers (position numbers in the range 1 to 100) then print the values stored in the elements between the two positions. (eg an input of 3 7 should cause the printing of the contents of a[3], a[4], a[5], a[6], a[7])

```
input start, finish
for n=start to finish
   print a[n]
endfor
```

4.12 Input 2 values - then look at the value stored in each element of the array (100 of them) and print each value that falls between the 2 input values

```
input low, high
for n=1 to 100
if (a[n]>=low) and (a[n]<=high)
   print a[n]
   endif
endfor
```

4.13 100 students sit a 'before' exam, and their marks can be assumed to be in the array b. Later, they do an 'after' exam, and those marks are in a. Print the subscript number (i.e in range 1 to 100) of the student who:

a) got the lowest mark overall (before + after)
b) improved the most between b and a.

```
a).
   lowposition=1
   lowvalue=1
   for student=2 to 100
      if a[n]+b[n] <lowvalue
         lowvalue=a[n]+b[n]
         lowposition=n
      endif
   endfor
   display lowposition

b).
   improveposition=1
   improvemost=1
   for student=2 to 100
```

```
   if a[n]-b[n] >improvemost
      improvemost=a[n]-b[n]
      improveposition=n
   endif
endfor
display improveposition
```