STANDARDS FOR WRITING ALGORITHMS

Everyone Please use the following standards below when writing your algorithms.

- Algorithms will be written in structured English statements, meaning that it will be a mix between mathematical notation and English statements. Also keep in mind that indentation is very important when writing algorithms.

Use the following guidelines when writing your algorithms:

1. Statements should be clear and unambiguous
2. Use one line per logical element
3. All logic should be expressed in operational, conditional, and repetition blocks
4. Logical blocks should be indented to show relationship
5. Keywords should be capitalized

Here are some common keywords used in algorithms:

START, BEGIN, END, STOP, DO, WHILE, DO WHILE, FOR, UNTIL, DO UNTIL, REPEAT, END WHILE, END UNTIL, END REPEAT, IF, IF THEN, ELSE, IF ELSE, END IF, THEN, ELSE THEN, ELSE IF, SO, CASE, EQUAL, LT, LE, GT, GE, NOT, TRUE, FALSE, AND, OR, XOR, GET, WRITE, PUT, UPDATE, CLOSE, OPEN, CREATE, DELETE, EXIT, FILE, READ, EOF, EOT, LOOP, ARRAY, SWAP, PROCEDURE, FUNCTION, RETURN …

Declaring Variables:

- In algorithms notation we don't declare variables we are going to use. The variables however can be initialized before being used. To initialize a variable you place the variable name on the left hand side followed by := then the value.
    - Examples: x := 1, y := 2, z := 3;

Arrays

- To declare an array you use the keyword "array" followed by the array name and length of array in brackets.
    - Example: **array** apples[1…n]

```
array x[2…100]
array firstnames[0…(n-1)]
```

- To access the index of an array you use the bracket notation.
    - Example: `array x[1…n]`

```
x[3] := 0
```

Matrix Array

- Example of matrix array:

```
matrix: array [1…20, 1…20] of integers
*note: matrix is the name of the array. The "of"
keyword is used to indicate the type of indexes
there are, in this case they are integers.

names: array [1…10, 1…30] of strings
```

- To access index of matrix array you use the bracket notation as well but with a comma to separate the dimensions.
    - Example: `matrix[1,4]`

```
names[3,1]
```

Functions and procedures

- The main difference between a function and a procedure is that a function returns a value while a procedure does not.
    - Example of procedure:

```
procedure swap(x, y)
     temp := x
     x := y
     y := temp
```

- Example of function

```
function add1(x)
     x := 1 + x
     return x
```

*note the indentation of the return statement it is not lined up with the function on purpose its indented…

Loops

- To perform a loop of any sort you need to be clear when the loop breaks or when it returns. Keywords that are commonly used are repeat, for, while.

  - Examples of repeat x times :

```
repeat n-2 times
     x := x + 1
```

  - Example of repeat until:

```
repeat
     j := j-1
until x > 2 or j = 1
```

  - Examples of while loop:

```
While m > 0 and n not equal to 5 do
     m := m-1
     n := n + 1
```

  - Example of for loop:

```
for i := 1 to m do
     m := m mod i

for i := 5 to m step 2 do
     write i*m + 1
```

Integer division and normal division

- To use integer division you use the "÷" symbol. To indicate that you are using normal division you use the "/" symbol

Leaving comments

- To leave a comment in your algorithm you encapsulate your comment in curly braces.
  - Example: {This procedure assumes m > 1}

Algorithm examples:

```
procedure pigeonhole( T[1…n] )
     {this procedure sorts integers between 1 and 10000}
     array u[ 1…10000]
     for k:= 1 to 10000 do u[k] := 0
     for i := 1 to n do
```

```
            k := T[i]
            u[k] := u[k] + 1
      i := 0
      for k := 1 to 10000 do
            while u[k] not equal to zero do
                  i := i + 1
                  T[i] := k
                  u[k] := u[k] - 1
```

*Note that we do not return anything in this procedure

```
function Fibonacci (n)
      {Fibonacci calculates the n-th term of the Fibonacci
      sequence}

      i := 1 and j := 0
      for k := 1 to n do
            j := i + j
            i := j - i
      return j
```

*Note the return statement in this function

Example of recursion in algorithms:

```
Function fib (n)
      If n < 2 then return n
      else return fib( n - 1 ) + fib( n - 2)
```

Finally a useful tool to writing algorithms is called LaTeX. All the lab machines have latex installed. Here are good examples of how to install and use latex for algorithms.

- Examples of how to use LaTeX for algorithms pseudo code.
  - http://en.wikibooks.org/wiki/LaTeX/Algorithms_and_Pseudocode
- LaTex file for algorithmics pseudo code
  - http://texcatalogue.sarovar.org/entries/algorithms.html
- LaTex installation and documentation
  - http://mirror.unl.edu/ctan/macros/latex/contrib/algorithms/algorithms.pdf