BALG PROGRAM

BIGHT ALGORITHM

(latest version at the time of writing is 2009 February 1rst)

This one will compute, after you entered Number of LEAD (no limitation is written in the command lines) and the Number of BIGHT (again no limitation…..)

The only control is for the GDC rule. (Greatest Common Divisor : see my articles on MATHEMATICS OF THK)

So

INPUT

--- Number of LEAD (ex : 7)
--- Number if BIGHT (ex : 9)

OUTPUT (in the order from top to bottom they will be put on the STACK)

--- DELTA (ex : 4)
--- DELTA* (ex : 5)
--- (L)mod B (ex : 7)
--- (-L)mod B (ex : 2)
--- Complementary sequence (ex : { 0 5 1 6 2 7 3 }+++
--- Periodic sequence (ex : { 3 7 2 6 1 5 0 }+++

This is sufficient for you to manually , using paper and pencil, compute the coding of each half-period of the knot
+++ you will wish to avoid going HALL's way !
DO NOT SUPPRESS THE '0' , this is essential to disambiguation and swift orientation, plus it is intellectually more coherent with the knot itself.
These '0' are – as all '0' of good lineage are – just "place holder" ; they stand for the BIGHT RIM which have no crossing so they will not be actively used but are "passively essential to understanding"

READ SCHAAKE and TURNER
READ my TURKSHEAD pages on this

Just a short summary of how to do (not a great deal on the essential "why" as this seems to go over too many heads interested only in "results") :
7L 9B REAL THK (see my articles THK or NOT THK and MATHEMATICS and THK)

9 BIGHT (to be numbered not from 1 to 9 but from 0 to 8) so put on nine marks, here x

x x x x x x x x x

Above the **left**most 'x' mark write the BIGHT RIM place holder '**0** '

0 . . . .
x x x x x x x x x

Now there are 8 '**.**' missing their digit : the digit to be used are those left or 1 to 8.

How to proceed to put them in the place they belong to ?

You can either go the slow way and use

(**-**L)modB or as I prefer as it avoid an

error prone '**-**' just before the L :

(B - L)mod B

(-7)mod 9 = 2 (9-7)mod9 = 2

This will give you your "step" to walk the X X X X plank

Remember this is 'circular' or 'modular"

0 . 1 . . . . . .
x x x x x x x x x

----

0 . 1 . 2 . . . .
x x x x x x x x x

----

0 . 1 . 2 . 3. . .
x x x x x x x x x

----

0 . 1 . 2 . 3. . 4
x x x x x x x x x

----

0 5 1 . 2 . 3. . 4
x x x x x x x x x

----

0 5 1 6 2 . 3. . 4
x x x x x x x x x

----

0 5 1 6 2 7 3 8 4
x x x x x x x x x

Or go the direct way at the price of computing the value of DELTA*

Which will give you the "direct writing increment" ; in th example it is 5

0 5 10 15 20 25 30 35 40
x x x x x x x x x

only problem is that we are only "allowed" 0 to 8 !

What have we done wrong ?

We 'just' forgor "circular', 'modular' and did not apply the MODULO to our results.

5 mod9==5 10 mod 9==1

15 mod 9==6 20 mod 9==2

25 mod9==7 30 mod 9==3

35 mod9==8 40 mod 9==4

hence

0 5 1 6 2 7 3 8 4
x x x x x x x x x

this is for the **COMPLEMENTARY** BIGHT sequence,

now we need the **PERIODIC** BIGHT sequence.

There is a non computational way, practical way to get it :

READ the COMPLEMENTARY as it was written FROM LEFT TO RIGHT

Now WRITE FROM RIGHT TO LEFT what you are reading.

You are READING

0 5 1 6 2 7 3 8 4

so you will be WRITING

								0
						1	5	0
						1	5	0
					6	1	5	0
4	8	3	7	2	6	1	5	0

{ 4 8 3 7 2 6 1 5 0 } is the **PERIODIC** BIGHT sequence

Nice but what do we do with that ? Well…..we……..put it to good use ! ;-)

Put a line of @ marks , as much as there are LEAD plus ONE

7 LEAD 9 BIGHT

@ @ @ @ @ @ @ @

The @ stands for the **LEFT** side BIGHT RIM (mandrel frame of reference ; for cylinder it is Bottom Bight Rim). BIGHT RIM == NO CROSSING TO BE FOUND THERE.

The @ stands for the **RIGHT** side BIGHT RIM (mandrel frame of reference ; for cylinder it is Top Bight Rim). BIGHT RIM == NO CROSSING TO BE FOUND THERE.

So

@ will received the “0” place holder ABOVE and **NOTHING** UNDER

@ will received the “0” place holder UNDER and **NOTHING** ABOVE

ABOVE = you write the COMPLEMENTARY
UNDER = you write the PERIODIC

COMPLEMENTARY IS WRITTEN AND READ LEFT TO **RIGHT IN THE USUAL WAY**
PERIODIC IS WRITTEN AND READ RIGHT TO **LEFT IN THE ARABIC WAY**

This comes with the way S & T decided to draw their diagram for a knot on a mandrel:
The BIGHT RIM are on the left and on the right ; beginning the knot with a first HALF-PERIOD (all the ODD H-P) going up with a slant : low left to up right so **LEFT** TO **RIGHT** and the second
H-P (all the EVEN H-P) going up with a slant from low right to up left so **RIGHT** TO **LEFT** .
We are writing and reading as the Wend is reading the numbers.
See one diagram in S & T)

So now we have :

	0	5	1	6	2	7	3	**8**	**4**
	@	@	@	@	@	@	@	@	
4	**8**	3	7	2	6	1	5	0	

There is a “slight” problem : there is something where there should be nothing.
Simple enough : erase or don not write it down.

0	5	1	6	2	7	3	
@	@	@	@	@	@	@	@
	3	7	2	6	1	5	0

Note : Had we be confronted with something like a complementary bight *sequence* of 5 but the need of a complementary bight *algorithm* of 12

13L 5 B

COMPLEMENTARY { 0 3 1 4 2 }
PERIODIC { 2 4 1 3 0 }

0	3	1	4	2									
@	@	@	@	@	@	@	@	@	@	@	@	@	@
									2	4	1	3	0

We are missing some “material” for allocation to the empty places.
We just “make do by “adding” (complying with ‘direction’) sequences as needed

Adding is made **left** to **right**

0	3	1	4	2	0	3	1	4	2				
@	@	@	@	@	@	@	@	@	@	@	@	@	@
				2	4	1	3	0	2	4	1	3	0

Adding is made **right** to **left**.

Not enough ? we add again but only what we need, we don’t want to be greedy !

0	3	1	4	2	0	3	1	4	2	0	3	1	
@	@	@	@	@	@	@	@	@	@	@	@	@	@
	1	3	0	2	4	1	3	0	2	4	1	3	0

Simple , no ?

We are not out of the wood yet !

We have to add the crossing SPECIFYING THEIR NATURE (OVER / UNDER or HIGH / LOW) AS SEEEN BY THE Wend TRAVELLING THE HALF-PERIODS.
The UNDER of one(ODD H-P) is the OVER of the other (EVEN H-P) and vice versa.
So for the sake of generality we will avoid using the centre row of coding such as
‘ / \ / \ / \ ‘
We put in the **SEQUENCE OF CROSSINGS AS SEEN BY THE FIRST HALF PERIOD (or any ODD H-P) IN A FINISHED KNOT**

There are as much of crossing in a H-P as there are LEADS MINUS ONE

So for 13L we need 12 crossings

	O	U	O	U	O	U	O	U	O	U	O	U	
0	3	1	4	2	0	3	1	4	2	0	3	1	
@	@	@	@	@	@	@	@	@	@	@	@	@	@
	1	3	0	2	4	1	3	0	2	4	1	3	0
	U	O	U	O	U	O	U	O	U	O	U	O	

Now suppose we want the 8^{th} Half-Period coding. (remember the FIRTS H-P IS ALWAYS A FREE RUN DEVOID OF CROSSING so no need to compute it but the other need to be calculated)

8^{th} is **EVEN** so we make use if the first one of these formula
(Have no fear they will be put in a Program for the lazy ones ! – later)

EVEN H-P (HPE is the number of the H-P searched for)
I (Bight Index) = (HPE – 2) / 2

ODD H-P (HPO is the number of the H-P searched for)
I (Bight Index) = (HPO – 3) / 2

So for **8^{th} H-P**
I (Bight Index) = (HPE – 2) / 2 **I = (8 – 2) / 2 = 6 / 2 = 3**

for the **5^{th} H-P**
I (Bight Index) = (HPO – 3) / 2 **I=(5 – 3) / 2 = 2 / 2 = 1**

Let us see the 8^{th} H-P (**ODD** numbered it is, so **LEFT** TO **RIGHT** hence WE USE THE **COMPLEMENTARY** OR THE **UPPER PART** ABOVE THE LINE OF @ @ @)

Index value has been calculated as being **3**
All the 'Index' equal or less that 3 will "see" a crossing done : 3, 2, 1, 0

Remember we read **LEFT** TO **RIGHT**

	O	U	O	U	O	U	O	U	O	U	O	U	
0	3	1	4	2	0	3	1	4	2	0	3	1	
@	@	@	@	@	@	@	@	@	@	@	@	@	@
	1	3	0	2	4	1	3	0	2	4	1	3	0
	U	O	U	O	U	O	U	O	U	O	U	O	

or

O	U	U	O	U	O	O	U	O	U
3	1	2	0	3	1	2	0	3	1

We read
OVER-UNDER-UNDER-OVER-UNDER-OVER-OVER-UNDER-OVER-UNDER

H-P 8 === O1 U2 O1 U1 O2 U1 O1 U1

Let us see the 5^{th} H-P (**EVEN** numbered it is, so **RIGHT** TO **LEFT** hence WE USE THE **PERIODIC** OR THE **LOWER PART** UNDER THE LINE OF @ @ @)

Index has been calculated as being **1**

All the 'Index' equal or less that 1 will "see" a crossing done : 1, 0
Remember we read **RIGHT** TO **LEFT**

	O	U	O	U	O	U	O	U	O	U	O	U	
0	3	1	4	2	0	3	1	4	2	0	3	1	
@	@	@	@	@	@	@	@	@	@	@	@	@	@
	1	3	0	2	4	1	3	0	2	4	1	3	0
	U	O	U	O	U	O	U	O	U	O	U	O	
	Last read										**first** read		

UNDER-OVER-OVER-UNDER-UNDER

So H-P 5 == U1 O2 U2

Do your home work and find the other H-P coding

First-HP is FREE RUN
BUT ….. you have to know how many half-turn wrap are to be done between you starting pin on the left Bight rim and your arrival pin on the right Bight rim.
In this case it is 2 half-turn or one 360° / 2 Pi full turn

You get that by L / B
or 13 / 5 = 2 plus a remainder of 3 or 2.6 use the IP (Integer Part) and leave the FP (Fraction Part) so you use 2 and leave .6

H-P 1== FREE RUN with a wrap of 2 half-turn
H-P 2 == O1 U1
H-P 3 == O1 U1
H-P 4 == U1 O2 U1 U1
H-P 5 == U1 O2 U1 U1
H-P 6 == U2 O3 U2
H-P 7 == U2 O3 U2
H-P 8 == O1 U2 O1 U1 O2 U1 O1 U1
H-P 9 == O1 U2 O1 U1 O2 U1 O1 U1
H-P 10 ==O1 U1 O1 U1 O1 U1 O1 U1 O1 U1 O1 U1

Of course we could have chosen any coding as seen by the first H-P IN THE FINISHED KNOT as long as this coding was COLUMN coded or ROW AND COLUMN CODED.

(A particular procedure is to be used for LONG(L-B>>2) knots made on a THK SHADOW or CORDAGE ROUTE (string run for S & K) ex 101L 8B). Do you imagine yourself using a 100 long Bight Algorithm made by adding 8 long Bight sequence following adding 8 long Bight sequence ?

The way to get the code for each H-P is different with ROW coded and Neither ROW NOR COLUMN coded.

Using SCHK program (put in PGR with a SOBRE O1 – U1 coding , and in PRG3 you may chose the coding ex a non-sobre U1-O1 or anything you like complying with the at least COLUM coded condition) will do all the hard work for you but I believe you NEED to fully grasp the inside of the computation.

For the ROW coded use SCHR and for all (Column, Row, Row AND Column, Neither/Nor) JOAT or BINX .
For the STANDARD HERRINGBONE-PINEAPPLE use PINAPL or PINAPL2 .

Even the poor souls that have been unable to master the EMU48 and my programs are now with enough tips to usefully use SCHAAKE creation.