

C SC 230 – Fall 2010 – Assignment 4

Due Friday Aug. 3 by 13:00 in the course box in ECS, but will be accepted (without penalty) until Wednesday Aug. 8 at 13:00

Total Marks = 50

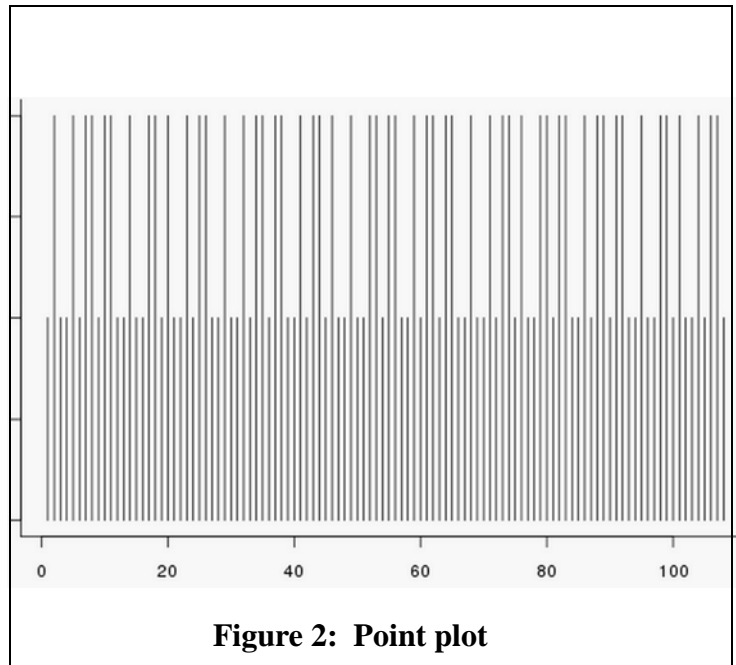
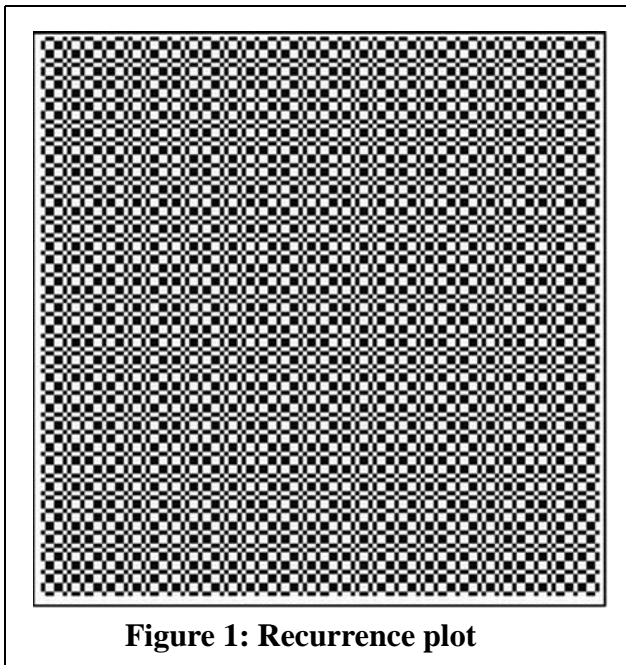
PART 1: The Problem to be solved: the Kolakoski Sequence

The Kolakoski sequence (A006928) is an infinite list whose initial 108 entries are shown in Table 1.

Table 1: The first 108 entries of the Kolakoski sequence A006928

<i>elements 1 to 18</i>	1	2	1	1	2	1	2	2	1	2	2	1	1	2	1	1	2	2
<i>elements 19 to 36</i>	1	2	1	1	2	1	2	2	1	1	2	1	1	2	1	2	2	1
<i>elements 37 to 54</i>	2	2	1	1	2	1	2	2	1	2	1	1	2	1	1	2	2	1
<i>elements 55 to 72</i>	2	2	1	1	2	1	2	2	1	2	2	1	1	2	1	1	2	1
<i>elements 73 to 90</i>	2	2	1	2	1	1	2	2	1	2	2	1	1	2	1	2	2	1
<i>elements 91 to 108</i>	2	2	1	1	2	1	1	2	2	1	2	1	1	2	1	2	2	1

The only numbers allowed are 1 and 2. Each number states how many numbers need to be appended to the sequence. There can be no more than two of the same number in sequence. Every time a new number is considered (read), one must alternate between writing 1 and 2. A recurrence plot of the Kolakoski sequence is illustrated in Figure 1 and its point plot in Figure 2.



There are actually a few permutations of the Kolakoski sequence (with different number labels).

The easiest english explanation is found for the development of the Kolakoski sequence A000002, shown by example in Table 2.

Table 2: Example for Kolakoski sequence A000002

write 1;	==> [1]
read 1 as the number of 1's to write before switching to 2;	==> [1]
write 2;	==> [1 2]
read 2 as the number of 2's to write before switching back to 1;	==> [1 2 2]
read the new 2 as the number of 1's to write;	==> [1 2 2 1 1]
read the new 1,1 as stating one needs to write one 2's and one 1's	==> [1 2 2 1 1 2 1]
<i>continue generating forever.</i>	

Kolakoski's self-generating sequences are of interest to both computer scientists and mathematicians (see “A New Kind of Science”, p. 895, by Stephen Wolfram). One might expect that the density of 1's and 2's is 50% for each, but this conjecture has never been formally proved (see the On-Line Encyclopedia of Integer Sequences™ (OEIS™) at <http://oeis.org/Seis.html>). Further references can be found in N. J. A. Sloane and Simon Plouffe, “The Encyclopedia of Integer Sequences”, Academic Press, 1995.

1.1 Algorithm, PseudoCode and Output

While the steps in Table 2 appear to be a simple enough explanation, it is still far from being a proper algorithm ready for programming. It would be a great learning experience to leave the development of the algorithm and pseudo code as part of this assignment, but it may take away from the real focus, which is to program in both C and ARM and link the code together.

Thus the pseudo code is given to you in Figure 3 for the production of the Kolakoski sequence A006928 for which a closed form algorithm can be found (not for the one shown in Table 2 which is the A000002 sequence used only for explanation) . Please note carefully the language (in case you do not have much experience). When it is stated that the range of a variable is “from x to y” this implies that both x and y are included in the possible values.

```

Let Kseq be the the the array to contain the Kolakoski sequence
Initialize the beginning elements as:
    Kseq[0] is not used
    Kseq[1] = 1
    Kseq[2] = 2
Let n be the number of “run” steps in producing the sequence
where n ranges from 2 to infinity. In a practical program, n should range
from 2 to a given maximum number of “runs”, here denoted by MAXRUNS

    for (n=2 to MAXRUNS)
    {
        for(i=1 to i= Kseq[n])
        {
            append the element (1+(n mod 2)) to Kseq
        }
    }

```

Figure 3:
Pseudo code
for producing
the Kolakoski
sequence

The output of a program should be the list of entries of the Kolakoski sequence up to a maximum number of runs, together with a count of the total number of entries, the count of the number of elements equal to 1 and the corresponding count of the number of elements equal to 2. A sample output for a given maximum run of n=50 is shown in Figure 4.

Jean Luc Picard V00123456 Kolakoski Program starts Kolakoski sequence of length 75 with 50 Runs Number of 1's is = 38 Number of 2's is = 37 <table> <tr><td>1</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>1</td><td>2</td><td>2</td></tr> <tr><td>1</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>1</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>1</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>2</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td><td>1</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table> Kolakoski Program ends	1	2	1	1	2	1	2	2	1	2	2	1	1	2	1	1	2	2	1	2	1	1	2	1	2	2	1	1	2	1	1	2	1	2	2	1	2	2	1	1	2	1	2	2	1	2	1	1	2	1	1	2	2	1	2	2	1	1	2	1	2	2	1	2	2	1	1	2	1	1	2	1	2	2	1							Figure 4: Sample output
1	2	1	1	2	1	2	2	1																																																																										
2	2	1	1	2	1	1	2	2																																																																										
1	2	1	1	2	1	2	2	1																																																																										
1	2	1	1	2	1	2	2	1																																																																										
2	2	1	1	2	1	2	2	1																																																																										
2	1	1	2	1	1	2	2	1																																																																										
2	2	1	1	2	1	2	2	1																																																																										
2	2	1	1	2	1	1	2	1																																																																										
2	2	1																																																																																

Make sure your program can work properly even if the maximum number of runs (MAXRUNS) is a different number, as, in marking, we may change your code and run it again with different values! **DO NOT**, however, make MAXRUNS a variable input from the user, or from a command line, or a parameter to a function (instead use EQU and #define).

1.2 The program design

The structure of the program must include at least two functions called by “main”, namely: “AppendKseq” and “PrintKseq”. Do not forget to add appropriate messages at the beginning and at the end of your program, and whenever output is expected. The interface of these two functions should be as follows (no changes allowed):

AppendKseq. int AppendKseq(int Kseq[],int index,int *Num1)

Description: Append the next elements in the Kolakoski sequence to the initial array.

Inputs: address of initial Kseq array, index = next position to be filled; Num1 = address of variable for the number of entries equal to 1.

Output: index = next position to be filled.

Side effects: update number of entries equal to 1 in its variable Num.

Expectation: Kseq array has been initialized for the first 3 entries at least (i.e. index > 2)

PrintKseq. void PrintKseq(int Kseq[],int index)

Description: Print the Kolakoski sequence.

Inputs: address of Kseq array, index = next position to be filled.

Output: none.

Side effects: none. However the elements of Kseq are printed in rows.

Expectation: Kseq is not empty.

PART 2: The first deliverables: flowcharts

(a) Draw a precise and correct flow chart, call it FLC, which includes ***all*** portions of your program and shows the modular design. FLC will be your guide towards implementing this program in C.

- (b) Draw a precise and correct flow chart, call it FLA, which includes ***all*** portions of your program and shows the modular design. FLA will be your guide towards implementing this program in ARM.
- (c) FLC and FLA are part of your final submission (on paper).

PART 3: The implementation using C

Following your design from FLC, implement your program using C, in a file called “A4KolakoskiC.c”, compiled using:

```
gcc -Wall A4KolakoskiC.c
```

Test your code for any number of MAXRUNS from 10 to 100 simply by recompiling it with different initial values (you must not read values from a file or from the command line or from `stdin`). Submit the program with a pre-encoded value of MAXRUNS=50.

PART 4: The second implementation using ARM

Following your design from FLA, implement your program using ARM, in a file called “A4KolakoskiA.s”, tested using ARMSim#. Test your code for any number of MAXRUNS from 10 to 100 simply by reassembling it with different initial values (you must not read values from a file or from `stdin`). Submit the program with a pre-encoded value of MAXRUNS=50.

PART 5: The third implementation using multiple files in ARM

It is good to get experience in compiling, assembling and executing programs which are split into multiple files. The important issue is to make sure that the separate module are interfacing properly and can communicate. Here you will place the function “AppendKseq” in a separate file such that it will be called as an external one by the main function in your program. The structure should become as shown in Figure 5:

<p>@ File: A4KolakoskiA-withExtern.s</p> <pre> [a] code and comments as before [b] .global _start [c] .extern AppendKseq [d] .global MAXRUNS [e] .text _start: code as before [f] without AppendKseq [g] .data code as before [h] .end </pre>	<p>@ File: A4AppendKseqA-Extern.s</p> <pre> code and comments as before [i] .global AppendKseq [j] .text @int AppendKseq(..parameters..) [k] code as before [l] .data code as before [m] .end </pre>
--	---

Figure 5: Structure of the program divided into multiple files

The program starts from file “A4KolakoskiA-withExtern.s” (left hand side of Figure 5) where the code for the main routine and for PrintKseq remains. The code for AppendKseq has been moved to another file called “A4AppendKseqA-Extern.s” (right hand side of Figure 5). The connection between the files must be stated explicitly through the use of symbols being *exported* or *imported* as necessary, using the `.global` and `.extern` commands. At [b] the label `_start` is exported through the use of the `.global` Assembler command as usual. This signals to any other portion of the program where the entry point is as it must be unique. At [c] the command `.extern` states that the label `AppendKseq`, when invoked, will be found externally to this file (hopefully in a module jointly assembled). At [d], the label `MAXRUNS` is similarly exported as the function `AppendKseq` will need to use it. Note the difference in use between `.global` and `.extern`. The rest of the program at [f] remains the same except that the portion for the function `AppendKseq` is no longer here. Note that the parts for `data` and `end` at [g] and [h] are the same.

The file “**A4AppendKseqA-Extern.s**” contains the function `AppendKseq` and thus its name (label) is immediately exported for all to see at [i] through the `.global` command. The rest of the code for `AppendKseq` remains the same in [j] to [m], and it can even have its own data (not necessary here) and it must have its own end command.

Once the `.global` and `.extern` declarations have been properly set, the program can be executed in ARM-Sim# by using the “File | Load Multiple” to load both assembly files, through a dialogue box. Step through the execution. Notice the transition between modules when `BL AppendKseq` is invoked.

PART 6: The fourth implementation using both C and ARM

Finally write the last version of the Kolakoski program using a mixture of C and ARM. Use the instructions posted on the website to combine multiple object files and to make sure that all parameters are appropriate (be careful about what the compiled C code may change!).

Note carefully: the goal here is that you do not write any new code. You should be able to insert your original C code for `AppendKseq` in a separate file called by the ARM program. We will check if any changes were made between the two versions!

Use the following template with functions and procedures as indicated.

Main: in ARM assembly, in a file `A4KolakoskiAC.s`.
AppendKseq: in C, in a file `A4AppendKseqCsub.c`.
PrintKseq: in ARM assembly, in the same file as main.
Any other function: in ARM assembly, in the same file as main.

Basically you are repeating the third implementation while substituting the C code for `AppendKseq` in an external C source file, `A4AppendKseqCsub.c`, containing the C source code for `AppendKseq`, which needs to be cross compiled to ARM code. Then ARM-Sim# will be able to link it in with the assembled `A4KolakoskiAC.s` and the program should run as it did both in the second full ARM implementation and the third one with multiple files.

PART 7: What to submit: all the deliverables

1. Submit neat copies of FLC and FLA in the course box in the 2nd floor of the ECS building (do not forget to write your name on the papers)..
2. Submit printouts in the 2nd floor of the ECS building of the source code for:
 - “`A4KolakoskiC.c`” (from implementation 1).
 - “`A4KolakoskiA.s`” (from implementation 2).
 - “**`A4KolakoskiA-withExtern.s`**” and “**`A4AppendKseqA-Extern.s`**”(from implementation 3).
 - “`A4KolakoskiAC.s`” and “`A4AppendKseqCsub.c`” (from implementation 4).
3. Submit the electronic copy of all the source code as above through the Assignment menu in the Connex course website.

7.1 References

- A web calculator for the Kolakoski sequence and others in general:
<http://members.chello.nl/k.ijntema/kolakoski.html>
<http://members.chello.nl/k.ijntema/>
- Wolfram MathWorld:
<http://mathworld.wolfram.com/>
- The On-Line Encyclopedia of Integer Sequences™ (OEIS™): use the A006928 sequence
<http://oeis.org/Seis.html>