Assignment 1: PRNGs

1 Implementation – Part 1

Implement a program `prngtest` that subjects a sequence of "random" binary numbers to the χ^2 tests with t-tuples discussed in class. Your program should read a sequence of characters "0" and "1" from standard input and print its evaluation (a real value in $[0, \infty)$, which gives the multiple of the critical value of the 99% confidence interval. In other words, an output of (near) zero means "perfectly" random (according to the test) and 1 corresponds to a probability of just 1% that the given sequence is random (according to the test). The output should be produced once the standard input stream ends (EOF). The parameter t is be the first argument to the application. Make sure your implementation scales to reasonably large values for t and works for sequences of arbitrary length.

2 Example – Part 1

```
$ echo 0101010101010101 | prngtest 1
0
$ echo 0001101100011011 | prngtest 2
0
```

3 Implementation – Part 2

You are to design and implement various PRNG generators. Each program is to print a sequence of "0" and "1" characters to `stdout`. The first argument to the program is an integer specifying the length of the sequence that should be generated. Implement the following PRNG generators:

- A generator `prngnot` that does not produce a (good) random sequence
- A generator `prngrand` using the `rand()` function (map the int *correctly* to binary)
- A generator `prngrecu` using the standard C++ recurrence $X_{n+1} = (aX_n + c) mod m$. Experiment with different values for a, c and m and

submit your code with the best values you could find. Describe your reasoning about the choice of parameters in a comment in the source code.

- A generator `prngyours` using a recurrence function of your own design. You should evaluate the function and describe your evaluation strategy and conclusions in a comment in the source code.

4 Example – Part 2

```
$ prngnot 4
0000
$ prngrand 2
01
$ prngrecu 10
0110010110
$ prngyours 10
0110010110
```

5 Submission

You must submit the implementations to your subversion repository to the directory `courses/comp3704/s2009/$USER/p1/`. Do not include generated files. The files submitted should be named as follows:

- `Makefile`
- `prngtest.c`
- `prngnot.c`
- `prngrand.c`
- `prngrecu.c`
- `prngyours.c`

You must check that the submitted code compiles by invoking `make`. Verify that the output of your program matches the expected output using your own testcases.