VJ Davey

CSci 426 Homework 2

Excercises 2.1.5 and 2.2.11

1. Exercise 2.1.5

   (a) **Except the case $m = 2$, prove that $a = 1$ cannot be a full period multiplier.**
   1 cannot be a full period multiplier because of the nature of the function

   $$g(x) = ax \mod m$$

   Using any initial seed will result in that same initial seed being repeated endlessly due to the identity property of multiplication. 1 times any value produces that same value, and the subsequent modulo operation wont do anything but produce a remainder which is going to be our same initial seed (given that $x_0 < m$). *e.g.*: If $x_0 = 3$ and $m = 13$, our resulting sequence will be: $\{3,3,3,3...\}$. There is no possible way for this to sequence to ever be full period.

   (b) **What about $a = m-1$?**
   If $a = m-1$, then

   $$g(x) = (m-1)x \mod m$$

   .
   Taking note of this, whenever we make use of the seed $x_0 = 1$, the function $g(x)$ produces a result which makes $x_1 = (m-1)$. Solving for the next element of the sequence, and using our x_1 gives us

   $$g(x) = (m-1)(m-1) \mod m = (m^2-2m+1) \mod m$$

   The product of $(m-1)(m-1)$ always results in a number which is 1 more than a multiple of m. Because of this, the sequence produced is an alternating series of numbers between 1 and $m-1$. Therefore, use of $m-1$ as a multiplier cannot be full period with the exception of the special cases $m = 2$ and $m = 3$.

2. Exercise 2.2.11 **Let m be the largest prime modulus less than or equal to $2^{15}-1$.**
   This number is 32,749. I discerned this by computing 2^{15} and then looking at a list of known primes. I chose the prime number of highest value less than 2^{15}

   (a) **Compute all the corresponding modulus-compatible full-period multipliers**
   The following numbers were found by creating a C program which makes use of the algorithms provided on pages 52 and 44 of the textbook. I can share this program if needed. These numbers are:

   2, 32, 128, 76, 237, 131, 53, 139, 115, 97, 23, 92, 191, 218, 165, 419, 33, 528, 314, 89, 861, 261, 150, 193, 30, 120, 6, 24, 57, 167, 668, 159, 65, 13, 52, 99, 321, 4678, 409, 5458, 72, 171, 151, 175, 35, 140, 7, 28, 448, 229, 156, 425, 207, 85, 17, 68, 317, 963, 292, 74, 16374, 270, 54, 84, 798, 117, 61, 244, 142, 182, 204, 82, 779, 337, 442, 155, 31, 124, 219, 617, 1309, 162, 249, 63, 2046, 227, 172, 98, 153, 145, 372, 1488, 202, 29, 116, 242, 79, 3274, 118, 189, 129, 257, 44, 176, 67, 268, 279, 606, 348, 394, 50, 10, 160, 380, 94, 376.

(b) **Comment on how this result relates to random-number generation on systems that support 16-bit integer arithmetic only**

I would think that this result means that there are a number of options for random number generation for systems that support 16-bit integer arithmetic only. Its still very limited compared to the 32 and 64-bit counterparts.