Robert Johns
September 23, 2013

CSCI 426: Simulation
Homework 2

2.1.9 (a) Verify that the list of five full-period multipliers in Example 2.1.6 is correct.

Solution: The following program was used to verify the list:

```
int main(void) {
        long p = 1;
        long a = 447489615;
        long x = a;
        long m = pow(2,31) - 1;
        while (x != 1) {
                 p++;
                x = (a*x)%m;
        }
        if (p == m - 1) {
                printf("yea\n");
        }
        else {
                printf("nay\n");
        }
        return 0;
}
```

(b) What are the next five elements in the list?

Solution: The following code was used to find the next five:

```
int gcd (long a, long b) {
       int c;
        while ( a != 0 ) {
                c = a; a = b%a;  b = c;
        }
        return b;
}
int main(void) {
        long i = 1;
        long a = 7;
        long x = a;
        long count = 0;
        long m = pow(2,31) - 1;
        while(x != 1 && count < 10) {
                if (gcd(i, m -1) == 1) {
                        printf("yea: x = %d, i = %d\n", x, i);
                        count++;
                }
                i++;
                x = (a*x)%m;
        }
        return 0;
}
```

The next five values are, where $x = 7^i \mod 2^{31} - 1$:

$x = 680742115, i = 23$
$x = 1144108930, i = 25$
$x = 373956417, i = 29$
$x = 655382362, i = 37$
$x = 1615021558, i = 41$

2.2.13 if $m = 2^{31} - 1$, compute the $x \in \chi_m$ for which $7^x \mod m = 48271$.

Solution: The following code, a slightly modified version of the program for **2.1.9(b)** was used to find the solution:

```
int gcd (long a, long b) {
         int c;
        while ( a != 0 ) {
                c = a; a = b%a;  b = c;
        }
        return b;
}
int main(void) {
        long x = 1;
        long a = 7;
        long y = a;
        long m = pow(2,31) - 1;
        while(y != 1) {
                if (gcd(i, m -1) == 1 && y == 48271) {
                        printf("x = %d\n", x);
                        return 0;
                 }
                 x++;
                 y = (a*y)%m;
        }
        return 0;
}
```

The program yielded the result: $x = 1116395447$

2.3.3 A fair coin is tossed once. If it comes up heads, a fair die is rolled, and you are paid the number showing in dollars. If it comes up tails, two fair dice are rolled and you are paid the sum of the two numbers showing in dollars. Let X be the amount won. Enumerate all possible values of $X$and use Monte Carlo simulation to estimate the probability of each.

Solution: The possible values, and the axiomatic probabilities of each, are:

1: $\frac{1}{2} * \frac{1}{6} = .083$
2: $\frac{1}{2} * \frac{1}{6} + \frac{1}{2} * \frac{1}{36} = .097$
3: $\frac{1}{2} * \frac{1}{6} + \frac{1}{2} * \frac{2}{36} = .111$
4: $\frac{1}{2} * \frac{1}{6} + \frac{1}{2} * \frac{3}{36} = .125$
5: $\frac{1}{2} * \frac{1}{6} + \frac{1}{2} * \frac{4}{36} = .138$
6: $\frac{1}{2} * \frac{1}{6} + \frac{1}{2} * \frac{5}{36} = .152$

7: $\frac{1}{2} * \frac{6}{36} = .083$
8: $\frac{1}{2} * \frac{5}{36} = .069$
9: $\frac{1}{2} * \frac{4}{36} = .055$
10: $\frac{1}{2} * \frac{3}{36} = .041$
11: $\frac{1}{2} * \frac{2}{36} = .027$
12: $\frac{1}{2} * \frac{1}{36} = .013$

The following program was used as a Monte Carlo simulation for the scenario (`Random()` uses the rng.c file provided):

```
int main(void) {
         int pos[] = {0,0,0,0,0,0,0,0,0,0,0,0};
        double count = 0;
        while(count < 100000000) {
                double tr1 = Random() * 2;
                int r1 = floor(tr1);
                if (r1 == 0) {
                        int tr2 = Random() * 6;
                        int r2 = floor(tr2);
                        pos[r2]++;
                }
                else {
                        double tr3 = Random() * 6;
                        int r3 = floor(tr3) + 1;
                        double tr4 = Random() * 6;
                        int r4 = floor(tr4) + 1;
                        int sum = r3 + r4;
                        pos[sum-1]++;
                }
                count++;
        }
        int j;
        for(j = 0; j < 12; j++) {
                double prob = pos[j]/count;
                printf("Probability of %d: %.5f\n", j+1, prob);
        }
        return 0;
}
```

The program yielded the following results, which support the axiomatic probabilities:

Probability of 1: 0.08331 Probability of 2: 0.09727 Probability of 3: 0.11111
Probability of 4: 0.12506 Probability of 5: 0.13888 Probability of 6: 0.15273
Probability of 7: 0.08333 Probability of 8: 0.06943 Probability of 9: 0.05554
Probability of 10: 0.04168 Probability of 11: 0.02776 Probability of 12: 0.01390