# $O$ COMP 215 Algorithms $\Theta$

## Lab #2

This work should, if possible, be completed during the lab and submitted electronically in a file called expmod.cpp before 11:59pm today. If you finish before the end of the lab, you can start the first assignment.

## Modular Exponentiation

Modular exponentiation – computing $b^n \bmod m$, $b$ exponent $n$ modulo $m$ – is one of the most fundamental operations used in modern cryptography (your computer makes at least one of these, with VERY large numbers, every time you establish an SSL channel - that small lock that appears in your internet browser).

In this exercise, you will program four function, expmod1, expmod2, expmod3 and expmod4, each implementing modular exponentiation in a different way. Two of them will be recursive, two will use a loop. Each function will take three unsigned long integers as input, a base $b$, an exponent $n$ and a modulus $m$, and output an unsigned long integer. And as a reminder, the "modulus" operation in C++ is done using the symbol `%` (so writing `x = y % z;` will put $y \bmod z$ in $x$).

Each function should test at the beginning that the base is an integer greater than 1 (otherwise the result is too easy), and that the modulus is smaller than 65,536 (otherwise, the computation require an integer larger than is allowed by an unsigned long). Also, if the base is larger than the modulus, replace the base by `b % m`. This check should be done only once, so you will need a helper function to keep this check out of the recursive functions.

1. The first function, expmod1, is a recursive function that uses the fact that if the exponent is 0, then the result is 1, otherwise, $b^n \bmod m = [b * (b^{n-1} \bmod m)] \bmod m$.

2. The second function, expmod2, initially sets a variable 'result' to 1 and then computes $b^n \bmod m$ using a loop which repeats $n$ times, and at each iteration, computes `result = (result * b) % m;`.

3. The third function, expmod3, is a recursive function that uses the fact that if the exponent is 0, then the result is 1, if the exponent is odd, then $b^n \bmod m = [b * (b^{n-1} \bmod m)] \bmod m$, and if the exponent is even, then $b^n \bmod m = (b^2 \bmod m)^{n/2} \bmod m$.

4. The fourth function, expmod4, initially sets the variables 'result' to 1, 'base' to $b$ and 'exp' to $n$, and then computes $b^n \bmod m$ using a loop which, while exp is not equal to zero, at each iteration, computes { `result = (result * base) % m; exp = exp - 1;` } if exp is odd, computes { `base = (base * base) % m; exp = exp / 2;` } if exp is even. (Bonus points to those who can prove why this computes the correct result.)

5. As a test, compute '$21415^{25000} \bmod 31457$', '$21415^{1,000,000} \bmod 31457$', '$21415^{100,000,000} \bmod 31457$' and '$21415^{1,000,000,000} \bmod 31457$' using each of your four function for each computation. What are you results? Do you run into any trouble? Can you imagine why? Do you notice any other notable difference between the functions?

6. Let's now try to compare the exact time taken by each function. Since modern computers are VERY fast, we will need to repeat the operations many times for differences to become clear, and you will need functions in `time.h` to measure the time taken as follows:

```
clock_t start,end;
start = clock();
/* do some stuff */
end = clock();
cout << "The computation took " << (end - start) << " clock ticks."
```

   Write four functions, time1, time2, time3 and time4, which each repeats the computation of $21415^{25000}$ mod 31457 fifty thousand times using expmod1, expmod2, expmod3 and expmod4 respectively. How many clock ticks did each take?

7. The previous attempt probably could not distinguish between expmod3 and expmod4, so write functions bigtime3 and bigtime4 that repeat the same computation, '$21415^{25000}$ mod 31457', but this time, ten million times with functions expmod3 and expmod4 respectively. (Don't try this with expmod1 and expmod2, I don't think they would have time to do this many repetition before the due date for this work!) How many clock ticks did each take?

**You can leave when:**

You have finished writing all the functions required above. The last three exercises require a written answer, write it in comments at the beginning of expmod.cpp that you submit electronically.