MACM 401/MATH 819
Assignment 3, Spring 2015.

Michael Monagan

Due Friday February 27th at 2pm.
Late Penalty: -20% for up to 70 hours late. Zero after that.
For problems involving Maple calculations and Maple programming, you should submit a printout of a Maple worksheet of your Maple session.

Question 1: The Fast Fourier Transform (30 marks)

(a) Let $n = 2m$ and let ω be a primitive n'th root of unity. To apply the FFT recursively, we use the fact that ω^2 is a primitive m'th root of unity. Prove this.

Also, for $p = 97 = 3 \times 2^5$, find a primitve 8'th root of unity in $\mathbb{Z}_p$. Use the method in Section 4.8 which first finds a primitive element $1 < \alpha < p-1$ of $\mathbb{Z}_p$. Then $\omega = \alpha^{(p-1)/n}$ is a primitive $n'th$ root of unity.

(b) What is the Fourier Transform for the polynomial $a(x) = 1 + x + x^2 + \cdots + x^{n-1}$, i.e. what is the vector $[a(1), a(\omega), a(\omega^2), \ldots, a(\omega^{n-1})]$?

(c) Let $M(n)$ be the number of multiplications that the FFT does. A naive implementation of the algorithm would lead to this recurrence:

$$M(n) = 2M(n/2) + n + 1 \quad \text{for } n > 1$$

with initial value $M(1) = 0$. In class we said that if we pre-compute the powers ω^i for $0 \leq i \leq n/2$ and store them in an array W, we can save half the multiplications in the transform so that

$$M(n) = 2M(n/2) + \frac{n}{2} \quad \text{for } n > 1.$$

By hand, solve this recurrence and show that $M(n) = \frac{1}{2} n \log_2 n$.

(d) Program the FFT in Maple as a recursive procedure. Your Maple procedure should take as input (n, A, p, w) where n is a power of 2, A is an array of size n created with $Array(0..n-1)$ storing the input coefficients $a_0, a_1, \ldots, a_{n-1}$, p a prime and w a primitive n'th root of unity in $\mathbb{Z}_p$. If you want to precompute an array $W = [1, w, w^2, \ldots, w^{n/2-1}]$ of the powers of w to save multiplications you may do so.

Test your procedure on the following input. Let $A = [1, 2, 3, 4, 3, 2, 1, 0]$, $p = 97$ and w be the primitive 8'th root of unity.

To see if your output B is correct, verify that when you apply the inverse FFT to B you get back A. Alternatively check $FFT(n, B, p, w^{-1}) = nA \mod p$.

(e) Let $a(x) = -x^3 + 3x + 1$ and $b(x) = 2x^4 - 3x^3 - 2x^2 + x + 1$ be polynomials in $\mathbb{Z}_{97}[x]$. Calculate the product of $c(x) = a(x)b(x)$ using the FFT.

If you could not get your FFT procedure from part (c) to work, use the following one which computes $[a(1), a(w), \ldots, a(w^{n-1})]$ using ordinary evaluation.

```
FFTfake := proc(n,A,p,w)
local f,x,i,C,wi;
    f := add(A[i]*x^i, i=0..n-1);
    C := Array(0..n-1);
    wi := 1;
    for i from 0 to n-1 do
        C[i] := Eval(f,x=wi) mod p;
        wi := wi*w mod p;
    od;
    return C;
end:
```

Question 2: The Modular GCD Algorithm (15 marks)

Consider the following pairs of polynomials in $\mathbb{Z}[x]$.

$$\begin{aligned}
a_1 &= 58\,x^4 - 415\,x^3 - 111\,x + 213 \\
b_1 &= 69\,x^3 - 112\,x^2 + 413\,x + 113 \\
a_2 &= x^5 - 111\,x^4 + 112\,x^3 + 8\,x^2 - 888\,x + 896 \\
b_2 &= x^5 - 114\,x^4 + 448\,x^3 - 672\,x^2 + 669\,x - 336 \\
a_3 &= 396\,x^5 - 36\,x^4 + 3498\,x^3 - 2532\,x^2 + 2844\,x - 1870 \\
b_3 &= 156\,x^5 + 69\,x^4 + 1371\,x^3 - 332\,x^2 + 593\,x - 697
\end{aligned}$$

Compute the GCD(a_i, b_i) via multiple modular mappings and Chinese remaindering. Use primes $p = 23, 29, 31, 37, 43, ...$ Identify which primes are bad primes, and which are unlucky primes. Use `Gcd(...) mod p` to compute a GCD modulo p in Maple and the Maple commands `chrem` to put the modular images together, `mods` to put the coefficients in the symmetric range, and `divide` for testing if the calculated GCD g_i divides a_i and b_i, and any others that you need.

PLEASE make sure you input the polynomials correctly!

Question 3: Resultants (15 marks)

(a) Calculate the resultant of $A = 3x^2 + 3$ and $B = (x-2)(x+5)$ by hand.

(b) Let A, B, C be non-constant polynomials in $R[x]$.
Show that $\text{res}(A, BC) = \text{res}(A, B) \cdot \text{res}(A, C)$.

(c) Let A, B be two non-zero polynomials in $\mathbb{Z}[x]$. Let $A = G\bar{A}$ and $B = G\bar{B}$ where $G = \gcd(A, B)$. Recall that a prime p in the modular gcd algorithm is unlucky iff $p|R$ where $R = \text{res}(\bar{A}, \bar{B}) \in \mathbb{Z}$. Consider the following pair of polynomials from question 4.

$$\begin{aligned}
A &= 58\,x^4 - 415\,x^3 - 111\,x + 213 \\
B &= 69\,x^3 - 112\,x^2 + 413\,x + 113
\end{aligned}$$

They are relatively prime, i.e., $G = 1$, $\bar{A} = A$ and $\bar{B} = B$. Using Maple, compute the resultant R and identify all unlucky primes. For each unlucky prime p compute the gcd of the polynomials A and B modulo p to verify that the primes are indeed unlucky.

Question 4: Division in $R[x_1, x_2, \ldots, x_n]$ (15 marks) (MATH 819 students only)

Let R be an integral domain and $A, B \in R[x_1, x_2, \ldots, x_n]$ with $B \neq 0$. We will develop a different algorithm for division of $A \div B$ based on the lexicographical monomial ordering.

(a) Let X, Y, Z be monomimals in $x_1, x_2, \ldots, x_n$. We will use $X >_{lex} Y$ to mean $X > Y$ in the pure lexicographical monomial ordering. Prove that $X >_{lex} Y \implies XZ >_{lex} YZ$ and use this to prove that $lm(AB) = lm(A)lm(B)$. Hence it follows that $lc(AB) = lc(A)lc(B)$ and $lt(AB) = lt(A)lt(B)$.

(b) Therefore if $B|A$ then $A = BQ$ for some quotient Q and $lt(BQ) = lt(B)lt(Q)$ hence $lt(B)|lt(A)$ and $lc(B)|lc(A)$ in R and the monomial $lm(B)|lm(A)$. And if $lt(B)$ does not divide $lt(A)$ then B does not divide A.

Let q be the quotient $lt(A)/lt(B)$. Then we can compute $C = A - Bq$ and proceed to test if $B|C$. Sketch an algorithm for dividing in $R[x_1, x_2, \ldots, x_n]$ and program it in Maple. Test your algorithm in $\mathbb{Z}[x, y, z]$ for the following input A, B.

$$B = xyz + 3\,x^2 - 2\,xz + 4\,yz - 3$$

$$Q = -3\,y^2z + 2\,xy + z^2$$

$$A := BQ$$

Note, you will need to compute $lt(A)$ in lexicographical order. The Maple command

```
> c := lcoeff(A,[x,y,z],'m');
```

computes the leading coeffient c and leading monomial m in lexicographical order with $x > y > z$.

Note, a difficult step in developing this algorithm is proving termination. In the normal division algorithm in one variable x the degree of the remainder polynomials is strictly decreasing. But here in $R[x_1, \ldots, x_n]$, even though we can show that $lt(C) <_{lex} lt(A)$ it is far from obvious that the division algorithm must terminate in a finite number of steps. A proof of termination is given in MATH 441.