Due at the start of class Thursday, March 29, 2012.

Problem 1. We are going to multiply the two polynomials $A(x) = 3 - 5x$ and $B(x) = 2 + 6x$ to produce $C(x) = a + bx + cx^2$ in three different ways. Do this by hand, and show your work.

(a) Multiply $A(x) \times B(x)$ algebraically.

(b) (i) Evaluate A and B at the three (real) roots of unity 1, i, -1. (Note that we could use any three values.)

(ii) Multiply the values at the three roots of unity to form the values of $C(x)$ at the three roots.

(iii) Plug 1, i, -1 into $C(x) = a + bx + cx^2$ to form three simultaneous equations with three unknowns.

(iv) Solve for a, b, c.

(c) (i) Evaluate A and B at the four (real) 4th roots of unity 1, i, -1, $-i$.

(ii) Multiply the values at the four 4th roots to form the values of $C(x)$ at the four 4th roots.

(iii) Create the polynomial $D(x) = C(1) + C(i)x + C(-1)x^2 + C(-i)x^3$.

(iv) Evaluate D at the four 4th roots of unity 1, i, -1, $-i$.

(v) Use these values to construct $C(x)$.

Problem 2. Use the FFT algorithm to evaluate $f(x) = 5 - 4x + 2x^2 + 1x^3 - 8x^4 - 4x^5 + 2x^6 + 3x^7$ at the eight 8th roots of unity mod 17. You may stop using recursion when evaluating a linear function $(a + bx)$, which is easier to do directly. The eight 8th roots of unity mod 17 are 1, 2, 4, 8, 16, 15, 13, 9; it is easier to calculate with 1, 2, 4, 8, -1, -2, -4, -8. Do this by hand, and show your work.

Problem 3. In order to obtain an efficient recursive algorithm to evaluate a polynomial of degree $n - 1$ at n points, it is important that, in each of the two recursive calls, both the degree of the polynomial is halved and the number of points is halved.

(a) (i) Give a recursive algorithm using A_{odd} and A_{even} that evaluates a polynomial of degree d at any n points. So the degree is halved but the number of points is not halved.

(ii) Write a recurrence for its running time. You may assume d is a power of 2.

(iii) Solve the recurrence (any way you like), but show your work.

(iii) How fast is it when $d = n - 1$?

(b) (i) Give a recursive algorithm that evaluates a polynomial of degree d at any n points, that keeps the degree fixed but halves the number of points in the two recursive calls.

(ii) Write a recurrence for its running time. You may assume n is a power of 2.

(iii) Solve the recurrence (any way you like), but show your work.

(iii) How fast is it when $d = n - 1$?