YONSEI UNIVERSITY

CSI3108-01 2015. 09. 21

Programming HW#3

(Divide-and-Conquer)

Max 40 points

Due on Oct. 2 (Fri) by 5pm

Implement FFT in the textbook for polynomial multiplication using Java. Input to your program is an arbitary polynomial $A(x)$ in the coefficient representation.

For evaluation, implement the function FFT in Figure 2.9. Assume that all the coefficients of a polynomial are *integers*.

Figure 2.9 The fast Fourier transform

```
function FFT(a, ω)
Input:  An array a = (a_0, a_1, ..., a_{n-1}), for n a power of 2
        A primitive nth root of unity, ω
Output:  M_n(ω) a

if ω = 1:  return a
(s_0, s_1, ..., s_{n/2-1}) = FFT((a_0, a_2, ..., a_{n-2}), ω^2)
(s'_0, s'_1, ..., s'_{n/2-1}) = FFT((a_1, a_3, ..., a_{n-1}), ω^2)
for j = 0 to n/2 - 1:
    r_j = s_j + ω^j s'_j
    r_{j+n/2} = s_j - ω^j s'_j
return (r_0, r_1, ..., r_{n-1})
```

Use the following equation for implementation of ω.

$$e^{2\pi i/n} = \cos\left(\frac{2\pi}{n}\right) + i \cdot \sin\left(\frac{2\pi}{n}\right),$$

where $i=\sqrt{(-1)}$, and the circular constant π have to be matched by 3.141592.

Input

The first line has the number of test cases. From the second line, each test case is given in a single line by ordering coefficients from the constant to the highest degree. Note that the highest degree of a polynomial is 63.

Output

For each test case, print '#test case number' in the first line. And then print n values; one value per line in the form of 'real part imaginary part', where n is the number of points.

Sample Input

20 // the no of test cases = 20
2 1 2 3 // highest deg=2, $1 + 2x + 3x^2$
5 -6 -5 -4 -3 -2 -1 // hightest deg=5, $-6 - 5x - 4x^2 - 3x^3 - 2x^4 - x^5$
7 0 1 2 3 4 5 6 7 // example in your text book
…

Sample Output

#1	#3
6.000000 0.000000	28.000000, 0.000000
-2.000000 2.000000	-4.000001 -9.656854
2.000000 0.000000	-4.000000 -4.000000
-2.000000 -2.000000	-4.000001 -1.656854
#2	-4.000000 0.000000
-21.000000 0.000000	-3.999999 1.656854
-4.707108 -8.949747	-4.000000 4.000000
-4.000000 -3.000000	-3.999999 9.656854
-3.292894 -0.949748	#4
-3.000000 0.000000	…
-3.292892 0.949747	
-4.000000 3.000000	
-4.707106 8.949748	