Implement FFT and its inverse

The problem asks you to write code to implement the recursive FFT algorithm and Inverse FFT algorithm. Your program should first take an input 'T' which is the number of test cases. Each test case contains two lines.The first line contains two integers 'op' and 'n' . If 'op' = 0 , perform FFT , otherwise perform Inverse FFT. , 'n' denotes the degree bound of the input polynomial. The next line contain 'n' elements . If op = 0 ,the input will be of the form a_0, b_0, a_1, b_1, ..., a_n-1, b_n-1. Here, the pair a_j, b_j will denote the complex number c_j = a_j + ib_j as the coefficient of x^j of the input polynomial. Note that n will be an integer, and a_j, b_j will be floating point numbers.

If op=1, the input that follows is y_0, z_0, y_1, z_1, ... y_n-1, z_n-1. The pair y_j, z_j specifies the complex number y_j + iz_j to be A(w^j) for some polynomial A, that is, it is the jth coordinate of a given DFT.

Update 1- (17/08/2017)

Make sure you output the numbers using this format , **"%.6lf %.6lf\n"** , (num.real_part,num.imaginary_part) . Note that the output below contains "-0.000000". This happens when you use the given format.

Use double instead of float. Use pi = acos(-1.0) for better precision.

We are now ingoring errors <=10^-3. Please submit your code again.

Constraints

Test cases = 100

n <= 2^14

-100 <= a,b,y,j <= 100

Example Input :

```
2

0 4

0 0 1.0 0 2.0 0 3.0 0

1 4

6.0 0 -2 -2 -2 0 -2 2
```

Example output:

```
6.000000 0.000000

-2.000000 -2.000000
```

```
-2.000000 0.000000

-2.000000 2.000000

0.000000 0.000000

1.000000 -0.000000

2.000000 0.000000

3.000000 0.000000
```

The first 4 lines of the output is the result of the FFT algorithm. Input = x + 2x^2 + 3x^3 . Ouput = [6, -2 - 2i, -2, -2 + 2i]

The next 4 lines of the output is the result of the inverse DFT algorithm. Input = [6, -2 - 2i, -2, -2 + 2i] , Output = x + 2x^2 + 3x^3