1. **(2 points each question) True or false?** Circle the correct answer. No explanation required. No points will be subtracted for incorrect answers, so guess all you want.

   F If $f(n) = O(g(n))$ and $r(n) = O(s(n))$ then $f(n) - r(n) = O(g(n) - s(n))$.

   T If $f(n) = O(g(n))$ then $f(n)^2 = O(g(n)^2)$.

   T Let $m = 2^k$ be the smallest power of 2 that is larger than n (i.e. $k = \lceil \log n \rceil$). Then $m = O(n)$.

   F In a depth first search of a *directed* graph, it is possible to have two vertices u and v with previsit and postvisit numbers: $(pre(u), post(u)) = (10, 30)$ and $(pre(v), post(v)) = (20, 40)$.

   T In a depth first search of a *directed* graph, if $(pre(u), post(u)) = (15, 18)$ and $(pre(v), post(v)) = (10, 20)$, then (u, v) is a back edge.

   F If $a^{N-1} = 1 \bmod N$, then N is a prime.

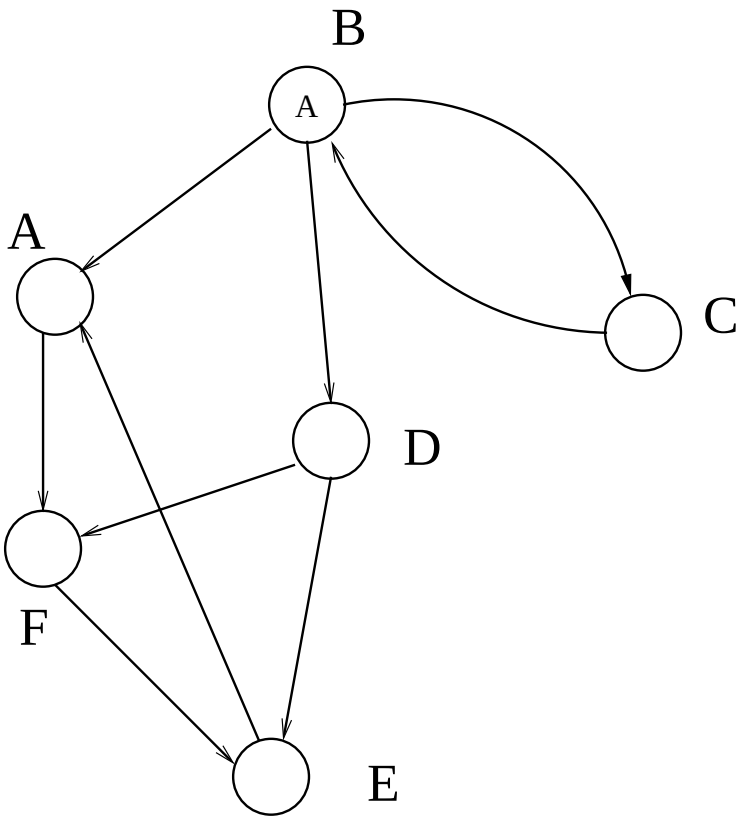
   T If $a^{N-1} \neq 1 \bmod N$, then N is composite.

   T If $a < 1$ then $1 + a + a^2 + \cdots + a^n = O(1)$.

   F If $a < 1$ then $1 + a + a^2 + \cdots + a^n = O(a^n)$.

   F There is no known algorithm for factoring N that runs in time $O(N^2)$.

2. (10 points) Run depth first search on the graph below, marking the post-visit time on each vertex. Whenever there is a choice, explore the vertex lower in alphabetical order.



The post-visit times are as follows: $A:6, B:12, C:9, D:11, E:4, F:5$.

3. (10 points) Evaluate $10^{25^{10}} \bmod 46$.

We will use $x^{ed} \equiv x \bmod N$, where $ed \equiv 1 \bmod (p-1)(q-1)$, so in this case $x^{1+22k} \equiv x \bmod 46$. $25^{10} \bmod 22 \equiv 1 \bmod 22$ since $a^{(p-1)(q-1)} \equiv 1 \bmod pq$ when $gcd(a, pq) = 1$. Then $10^{25^{10}} \equiv 10 \bmod 46 = 10$.

4. (15 points) Let $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ be a degree 3 polynomial such that $p(1) = 2, p(-1) = 0, p(i) = 1 - i, p(-i) = 1 + i$. Use the FFT method to find the coefficients a_0, a_1, a_2, a_3 of $p(x)$. Show your work.

   To find the coefficients, we notice that the polynomial is evaluated at points 1, ω, ω^2 and ω^3, where $\omega = i$ is the fourth root of unity. In order to find the coefficients we perform the inverse FFT transform: $FFT^{-1}(\omega) = \frac{1}{4}FFT(\omega^{-1})$

   The FFT inverse matrix is $FFT^{-1} = \frac{1}{4} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$

   To find the coefficients, we multiply the value vector $(2, 1 - i, 0, 1 + i)^T$ by the FFT inverse matrix. We obtain $(1, 0, 0, 1)$ so $p(x) = 1 + x^3$.

5. (25 points) Suppose that you are given the exact locations and shapes of several rectangular buildings in a city, and you wish to draw the skyline (in two dimensions) of these buildings, eliminating hidden lines. Assume that the bottoms of all the buildings lie on the x-axis. Building B_i is represented by a triple (L_i, H_i, R_i), where L_i and R_i denote the left and right x coordinates of the building, respectively, and H_i denotes the building's height. A skyline is a list of x coordinates and the heights connecting them arranged in order from left to right. For example, the buildings in the figure below correspond to the following input (the numbers in boldface type are the heights):

   $(1, \mathbf{5}, 5)$, $(4, \mathbf{3}, 7)$, $(3, \mathbf{8}, 5)$, $(6, \mathbf{6}, 8)$.

   The skyline (in bold) is represented as follows (again, the numbers in boldface type are the heights):

   $(1, \mathbf{5}, 3, \mathbf{8}, 5, \mathbf{3}, 6, \mathbf{6}, 8)$.

   (a) Given a skyline of n buildings and another skyline of m buildings, show how to compute the combined skyline for the $m + n$ buildings in $O(m + n)$ steps.

   Merge the "left" list and the "right" list iteratively by keeping track of the current left height (initially 0), the current right height (initially 0), finding the lowest next x-coordinate in either list; we assume it is the left list. We remove the first two elements, x and h, and set the current left height to h, and output x and the maximum of the current left and right heights.

   A common attempt was to merge the two lists according to x-coordinates while outputting the height of the corresponding skyline. (A variation was that in the case of a tie, the max of the two heights would be output.) This was worth 5 points as the heights are not dealt with properly.

   A better solution was to output set the current maximum height to the max of the next height, and output an x coordinate and the current height. This leads to a skyline where height only increases. This was worth 10-12 points depending on various details.

A full points alternative solution was to break up each skyline with the x-coordinates in the other skyline. Then one could compare the skylines in order, as they have the same x-cooredinates and outputting the maximum of each.

(b) Give a divide and conquer algorithm to compute the skyline of a given set of n buildings. Your algorithm should run in $O(n \log n)$ steps.

If there is one building, output it.

Otherwise, split the buildings into two groups, recursively compute skylines, output the result of merging them using part(a).

The runtime is bounded by the reccurrence

$$T(n) \leq 2T(n/2) + O(n),$$

which implies that $T(n) = O(n \log n)$.