CS 340 Programming Assignment IV:
Minimum Spanning Trees by Kruskal and Prim

Description: You are to implement **Kruskal's Algorithm** and **Prim's Algorithm** for finding Minimum Spanning Trees (MSTs) of undirected positively weighted graphs adhering to the specifications detailed below.

I/O Specifications: You will read your input graph from an input file named **graphin.txt** of the following adjacency list representation where each x_{ij} is the j'th neighbor of vertex i (vertex labels are 1 through n) and w_{ij} is the weight of the edge between vertices i and X_{ij}:

1: x_{11} w_{11} x_{12} w_{12} x_{13} w_{13}...
2: x_{21} w_{21} x_{22} w_{22} x_{23} x_{23} ...
.
.
n: x_{n1} w_{n1} x_{n2} w_{n2} x_{n3} w_{n3} ...

Your output from Kruskal's algorithm will be to a file named **kruskalout.txt**, and your output from Prim's algorithms will be to **primout.txt**, both of the following edge list form:

x_1 y_1
x_2 y_2
.
.
x_{n-1} y_{n-1}

This is just a list of the n-1 edges involved in the computed MST in the order of computation.

Algorithmic specifications:
Your algorithm must run in O(E log V) time on any input graph. Therefore, you must use the adjacency list representation for input processing. E.g., you may implement your adjacency list as an array of linked lists or an array of vectors. **For Kruskal's algorithm**, you must sort the edges by weight using an efficient (O(m log m)) sorting algorithm that you already implemented. In growing your forest of trees, you must implement an efficient Disjoint Set structure in which both the union operation and the find operation take O(log n) time, using either union-by-size or union-by-depth heuristics, stated in comments. **For Prim's algorithm,** you have already partially implemented a Heap and need to complete the implementation for a min-Heap including heap insertion, decrease key, and deletion, being careful to be able to access the heap positions from the adjacency list and vice versa.

What to Turn in: You must turn in a single zipped file containing your source code, a Makefile if your language must be compiled, appropriate input and output files, and a README file indicating how to execute your program (especially if not written in C++ or Java). **Refer to proglag.pdf for further specifications.**

This assignment is due by MIDNIGHT on Monday, March 12. Late submissions carry a -40 % per day late penalty.