You, an AI hater, recently learned that an AI model discovered new molecule structures equivalent to 800 years' worth of knowledge[1][2], and you are extremely annoyed by this news. While on an anger rant, you claim that you can write a basic program to find new optimal structures by yourself. Your chemist friends dare you to do so. They also offer to check the validity of your program, if you also write them another program that helps with their project.

You come up with an idea to find new molecule structures: If we have a graph where each vertex is an atom and each edge is the energy of the bond between the two atoms, maybe a new possible molecule structure can be found by selecting the edges with lowest energy bonds, without creating a cycle. Your chemist friends tell you this is most definitely a wrong approach, but you insist on trying.

Additionally, your chemist friends need a program to find the longest chain in a given molecule, so you agree to also write that program for them.

Problem

This exam consists of two parts, graded independently.

> PART1

You need to complete the function *find_stucture()* which returns the total bond energy of the found molecule:

```
int find_structure(std::vector< std::vector<std::pair<int,int> > >
&bond_energies, std::vector< std::vector<std::pair<int,int> > >
&lowest_energy_structure);
```

- *bond_energies*: a bidirectional graph represented by an adjacency list, where the vertices (v) are atoms, edges are bonds, and weights (w) of edges are the energies of the bonds. In other words, *an entry in the adjacency list for vertex v0 is <v1, w>, which represents an edge between v0-v1 with a weight of w.*
- *lowest_energy_structure*: the found molecule structure represented as an adjacency list, with the same format as *bond_energies*. You are expected to find the structure and assign it to this argument.
- *There is at most one bond with two atoms with only one weight value*, i.e. there is at most one undirected edge between any two vertices in the graph, meaning at most one weight value for each pair. *Undirectionality is shown in the adjacency list by adding two mirror entries for each edge* for ease of implementation.
- Vertices are represented as integers starting from 0, and the maximum number of vertices in the graph is 1000.

- Weights are represented as integers starting from 1, and the maximum weight value is 100.
- *return value* is the total bond energy of the *lowest_energy_structure*.

> PART2

You need to complete the function *find_longest_chain()* which returns the number of the atoms in the longest chain of the given molecule.

```
int find_longest_chain(std::vector< std::vector<std::pair<int,int> > >
&molecule_structure, std::vector<int> chain);
```

- *molecule_structure*: a graph with the same representation and limits as the *bond_energies and lowest_energy_structure arguments of PART 1. Additionally, it is guaranteed that the structure is a connected, acyclic, undirected graph, with all edges having the same weight (i.e. weights are not important).* Undirectionality is shown in the adjacency list by adding two mirror entries for each edge for ease of implementation.
- *chain*: a vector of integers, where each integer maps to a vertex ID in the found longest chain. The vector should follow the order of the chain. The validity of the chain will be checked by the tester to see if consecutive vertices in the vector have edges between them or not.
- *return value* is the *total number of atoms in the longest chain* of the molecule. The longest chain is defined as the count of vertices on the path between the two farthest vertices in the graph, including the start and end vertices.

Constraints and Hints:

- When deciding between edges with the same weights connecting v to two different vertices *u_1* and *u_2*, always favor the *u_i* with a smaller index.
- Similarly, if you need to decide between two edges *(u1,u3,w1)* and *(u2,u4,w2)* where *w1=w2*, you should again favor the edge with the smaller *u*_i, meaning (u1,u3,w1).
- There are no self-loops, meaning there is no edge such that *(u_i, u_j, w)* where *i=j*.
-