1. Goal: Accomplish the following two missions and integrate them into one single program. Deduct 5 points for unfriendly interface!

(Preprocessing) Input File & Build adjacency lists
Student Pairs (undirected graph): this binary file is named as pairs#.dat and sorted in ascending order of student id. The 3 attributes are as below.
- "sid1": an array of 16 characters for the first student.

- "sid2": an array of 16 characters for the second student.
- "weight": float, in (0, 1].

Adjacency Lists: Each "sid1" corresponds to an adjacency list. The nodes on an adjacency list are in ascending order of "sid2".
- The input file contains a variable number of pairs, so the space of your data structures MUST be dynamically allocated. A fixed-size array is NOT allowed!
- A node on adjacency lists MUST keep two pieces of information: "sid2" and "weight".

(Mission One) Find the minimum spanning tree in the maximal connected component
Input: the maximal connected component (with the highest number of students) found in the previous exercise.
Description: (1) Starting at the edge with the minimum weight, find the minimum spanning tree (MST) by Kruskal's algorithm and compute its sum of edge weights on MST.
(2) While expanding the minimum spanning tree, the edge with the minimal weight is selected first. Edges with equal weights are selected in ascending order of student id.
Ex. Assume that the latest-expanded edge is (v1, v2) and the four edges have equal weights. (v1, v3) < (v1, v4) < (v2, v3) < (v2, v4), so choose (v1, v3) first.
Output: Display on screen the sum of edge weights on the minimum spanning tree.

(Mission Two) Compute the shortest distance from the give student to each of the other students
Input: Adjacency lists & student id given by user.
Description: (1) Use Dijkstra's algorithm to compute the shortest distance from the given student to each of the other students on the graph.
(2) If there are more than one unvisited vertexes with the minimum distances, first choose the one with the smallest student id.
Output: Write "sid" and "shortest distance" of each student in the ascending order of "sid" as a file.

2. Documentation
The content must include but is not limited to the following:
(1) Tak Mission One as an example. Discuss the difference between Prim's and Kruskal's algorithms. Clearly point out which one is better and give a reason.
(2) How will you modify your program in Mission Two to compute the average of the shortest distances between every pair of two vertexes? Provide your result if you succeed; otherwise give a reason, please.

一、題目：完成下列兩項任務，並將兩者整合成單一程式，提供的操作介面若不友善先扣5分。

（前處理）讀檔及建立相鄰串列
學生配對(無向圖)：此二進位檔的檔名如pairs#.dat，每筆紀錄是依照對應的學號由小到大排序，共3個欄位如下：
-【學號sid1】第一位學生的學號以16個字元陣列表示
-【學號sid2】第二位學生的學號以16個字元陣列表示

- 【配對權重weight】以浮點數float儲存，介於0~1之間的正實數

相鄰串列：依【學號sid1】由小到大排序，每個【學號sid1】對應一條相鄰串列，各串列上依照【學號sid2】由小到大排序。

- 輸入資料是不固定的的筆數，資料結構必須以動態配置空間，直接用固定大小的陣列者，一律零分！
- 相鄰串列上的每一個節點必須存放資訊：【學號sid2】和【配對權重weight】。

（任務一）找出最大連通成分內的最小生成樹

輸入：沿用上一個練習中所找到的最大(學號筆數最多的)連通成分。

描述： (1) 以最大連通成分內權重最小的邊為起點，透過Kruskal's演算法找出最小生成樹，計算並輸出這棵樹上關係權重的總和。

(2) 展開最小生成樹下一個邊時，以關係權重最小者優先，權重相等時以擁有學號最小的邊優先。

例：目前已展開至{v1, v2}，假設可展開的4個邊權重都相等，則(v1, v3) < (v1, v4) < (v2, v3) < (v2, v4)，故先選(v1, v3)。

輸出：將最小生成樹的權重總和顯示於螢幕上。

（任務二）求出指定學生至每位學生的最短距離

輸入：相鄰串列和使用者指定的一個學號。

內容： (1) 透過Dijkstra's演算法逐一找出指定學生至圖上每位學生的最短距離。

(2) 若尚未走訪且目前距離最短的點超過一個，先選擇學號最小者。

輸出：依照學號由小到大的次序輸出每位學生的學號及其最短距離至檔案。

二、說明文件

內容必須包含但不限於以下幾項：

1. 探討Prim's和Kruskal's兩個演算法的差異，以任務一為例，明確指出哪一個演算法比較好並說明理由。
2. 若想知道任兩點之間最短距離的平均，如何改寫現有的任務二？請嘗試進行後提供成功獲得的結果，否則請說明原因。