1. Basic Requirements

(a) Only upload source codes in .cpp/.c/.h/.hpp with comments that can be successfully compiled, the file name should be "DS2ex5_*team-id_student-id1_student-id2*". Deduct 5 points immediately for any violation!

(b) Upload only one copy for each team and there must be the name and student id of each member at the first few lines in your codes. Deduct 5 points for duplicate or any missing information!

(c) Codes that are non-C/C++ or unable to be successfully executed will be treated as "Unfinished" and get no point.

2. Goal

Accomplish two missions and integrate them into one. Deduct 5 points for unfriendly interface!

(Mission One) Read a file to build adjacency lists

Input: Read a set of student pairs (undirected graph), a binary file taking pairs###.bin as its file name. Each record stands for a student pair with a non-zero weight. Their attributes are as follows:

- "sid1": an array of 10 characters for the first student.
- "sid2": an array of 10 characters for the second student.
- "weight": float, in the range of (0, 1].

Steps: Keep all the student pairs as adjacency lists in ascending order of "sid1", where each "sid1" corresponds to one adjacency list. The nodes on an adjacency list are sorted in descending order of "weight".

Adjacent Lists: 1. The input file has a variable number of pairs, so the space of your data structures MUST be dynamically allocated. A fixed-size array is NOT allowed!

2. Every node on adjacency lists MUST keep "sid2" and "weight".

Output: Write adjacency lists in ascending order of "sid1" as a text file with a name extension of .adj, including all the nodes on each adjacency list in descending order of "weight".

(Mission Two) Traverse a graph to find connected components

Input: (Only) The adjacency lists built in Mission One.

Steps: Start depth-first search (DFS) traversals at arbitrary nodes to find every connected component in the graph of student pairs.

Output: Write the connected components in descending order of their sizes as a text file with a name extension of .cc, including the student id's in ascending order in each connected component.

3. Flowcharts of two missions & Documentation

(a) It consists of two stages: two flowcharts during on-machine exercise, and a report for codes before the DEMO.

(b) Before the discussion board is closed, each team MUST have had a post in order to be arranged for the DEMO.

(c) The content must include but not limited to the following:

1. Introduction: Brief by text the main goal, assumptions, difficulties you encountered and the solutions. Do NOT copy anything directly from here.
2. Flowcharts: Insert one flowchart for each mission as a figure in your post.
3. Q&A: For what kind of applications can the program of finding connected components be useful? You MUST precisely describe the semantics of pair weights and the size of a connected component in it.

一、基本需求

(a) 只上傳可成功編譯的原始碼(.cpp/.c/.h/.hpp)含註解、檔名請用「DS2ex5_*分組編號_學號1_學號2*」，違反任何一項先扣 5 分！

(b) 以組為單位只上傳一份，程式碼開頭幾行註解必須要有整組每位同學的中文姓名和學號，多傳一份或資訊不完整就扣 5 分！

(c) 非 C/C++程式 或 無法成功執行 一律視為「未完成」並以零分計！

二、題目

完成兩項任務，將二者整合在一個簡易選單下，未整合或介面無法連續執行先扣 5 分。

（任務一）讀檔建立相鄰串列

輸入：讀入一個學生配對檔(無向圖)，此二進位檔以 pairs###.bin 為檔名，每筆紀錄代表非零權重的一組學生配對，包括 3 個欄位如下：

- 【學號 sid1】第一位學生的學號以 10 個字元陣列表示
- 【學號 sid2】第二位學生的學號以 10 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

步驟：依【學號 sid1】由小到大將所有的學生配對存成相鄰串列，每個【學號 sid1】對應一條相鄰串列，各串列上的節點依照【配對權重 weight】由大到小排序。

相鄰串列：(1) 輸入資料是不固定的筆數，資料結構必須以動態配置空間，禁用直接宣告固定大小的陣列！

(2) 相鄰串列上每一個節點必須存放【學號 sid2】和【配對權重 weight】。

輸出：依【學號 sid1】由小到大將每條相鄰串列寫入同檔名但改以 adj 為延伸檔名的文字檔，並依照【配對權重 weight】由大到小列出每條串列上的所有節點。

（任務二）走訪圖形找出連通成分

輸入：(僅限)使用任務一建立的相鄰串列。

步驟：從任意節點開始以 DFS 深度優先走訪相鄰節點，找出學生配對圖中彼此分離的每個連通成分。

輸出：依照所涵蓋的學號個數由大到小將所有連通成分寫入同檔名但改以 cc 為延伸檔名的文字檔，每個連通成分均由小到大列出所涵蓋的學號。

三、流程圖和程式說明文件

(a) 分為兩個階段，上機練習時要繳交兩張流程圖，機測前要繳交程式說明文件。

(b) 各組必須在看板關閉期限以前完成貼文，才會被排入機測。

(c) 貼文內容必須包含但不限於以下幾項：

1. 簡介：以文字簡述程式主旨，假設，遇到的困難和解法，勿直接剪貼題目字句！
2. 流程圖：每項任務各一張流程圖，以插圖放入貼文之中！
3. 答問：找出每個連通成分的程式有助於哪一種應用？須明確描述配對權重和連通成分大小在其中的意涵。