開發平台(Development Platform):win8.1

使用開發環境(Development environment):sublime text 3 with GCC

程式功能(Program Functions):

此程式可依助教的要求進行讀檔，並以檔案中的adjacency matrix，使用sollin 演算法建造並印出一minimum cost spanning tree。

程式設計(Program design):

<void ini\_matrix(void);>

初始化adj\_matrix，以接收數序

<int root\_find(int x);>

因為我以disjoint forest set的方式記錄每顆子樹，此函式可以尋找每個disjoint set的根值

<void tree\_union(int x, int y);>

使disjoint set中的兩顆子樹結合（將其中一個set串接於另一組set下）

<void ini\_minimum(void);>

初始化minimum\_link\_length陣列，此儲存特定子樹的最小link

<int judge\_tree\_num(int x);>

判斷當前的forest陣列所代表的disjoint set是否只包含一棵樹（也就是所連結的root值都一樣），若只包含一棵子樹則回傳一，沒有則回傳零。

<int main(void)>

首先會以fscanf來取得檔案內的資料，並且以一個二維陣列adj\_matrix接收（也就是這個二維陣列將會變為相鄰矩陣）。

再接收的同時判斷當前的邊長是否不等於零，若不等於則以edges結構陣列紀錄此邊的始點、終點以及邊長，並將forest陣列（其為disjoint set）初始化為每個節點的index（也就是子樹集為每個節點一棵子樹）。

接著以一迴圈執行以下步驟直到目前的forest陣列中只有一棵樹（以 judge\_tree\_num函式）。

每次尋找每個邊，判斷其邊的始點終點是否連接同一棵樹（也就是可否形成cycle，並且判斷此邊是否為已經生成的樹的連外邊，並且是否為連外邊中最小，若皆是則以minimum\_link\_length紀錄此連外邊的邊長、以minimum\_link紀錄此連外邊在edges陣列中的index

並且將所選到的連外邊加入到對應的子樹集合中，並以selected陣列紀錄該已經被選擇的邊。

最後，當所有子樹集都連在一起而成為單一一棵樹時（也就是迴圈的終止條件），迴圈將終止。

為了印出要求的輸出，我以雙層迴圈執行以下步驟：

1.以一迴圈尋找每次尋找當前最小生成樹的邊集合中值最小的一個，並印出其相關資訊（始點、終點、邊長）和將此邊從最小生成樹的邊集合中剔除，以進行下次搜索。

2.當連續執行節點個數-1次以上步驟後（也就是第二層迴圈），將會按照由小而大的順序印出所有最小生成樹中的邊