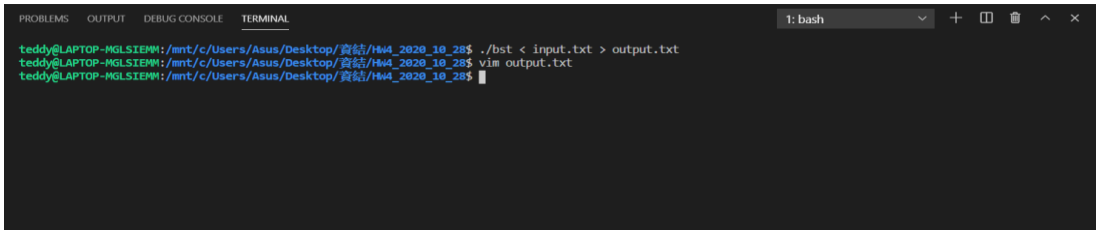
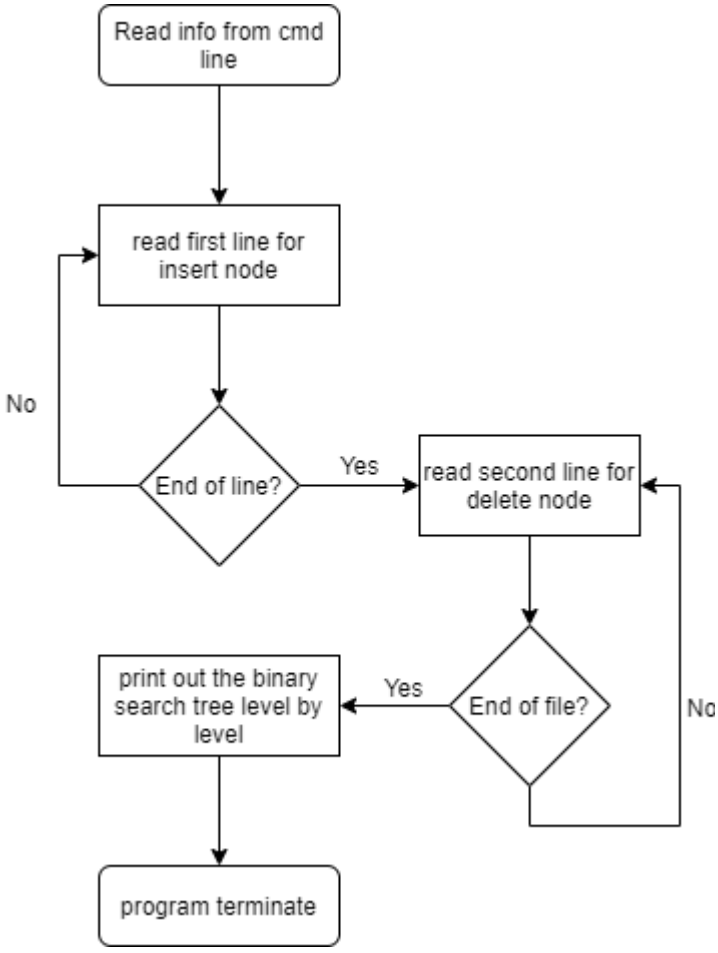
# F74076108_蔡秉睿_hw4_Readme

1. screen shot:





2. program architecture:




3. program function:
   i. node_t *newNode(int data):

      新增一個 node，用來當作 root。

Parameters:

int data: Node(root)的值。

ii. node_t *insert_node(node_t *node, int key):

新增一個節點，把它和 root 接起來，並依照 bimary search tree 排列。

Parameters:

node_t *node:要和新增 node 接起來的 root。

int key:新增的 node 的值。

iii. node_t *find_min_node(node_t *node):

找到這棵樹的最小值的 node。

Parameters:

node_t *node:這棵樹的 root。

iv. node_t *delete_node(node_t *node, int key):

刪除和 key 有相同值的節點，並根據 3 種情況(No child、One child、Two children)做後續處理。

Parameters:

node_t *node:要刪除節點的樹的 root。

int key:要刪除的節點的值。

v. int getLevelCount(node_t *node):

計算這棵樹的高度。

Parameters:

node_t *node:要計算的樹的 root。

vi. void printGivenLevel(node_t *node, int level):

print 出給定的 level(含)以後的值。

Parameters:

node_t *node:要 print 的樹的 root。

int level:要印的 level。

vii. void printout(node_t *node):

呼叫 int getLevelCount 和 void printGivenLevel，把 binary search tree 一層一層 print 出來。

Parameters:

node_t *node:要 print 的樹的 root。

4. Program design:

這次作業先利用 scanf 讀取第一行，然後把讀入的值做 binary search tree 的排列。再 scanf 第二行，把讀入的值從 tree 中刪除。Delete 中要判斷要刪除的節點是哪一種 case(No child、One child、Two children)。最後再把 tree 一層一層印出來。

這次遇到最大的問題是 scanf 怎麼判斷換行，找該怎麼判斷換行花了我最多

時間，我覺得 scanf 的判斷換行真的很難用。再來還有是 delete 的問題，因為我在判斷 3 種 case 時，一開始都把 return 都寫在各自的 if else 裡頭，而我又是用遞迴的方式來做，所以變成後面就沒東西 return，印出來的 tree 就變少或者是全部不見了。