Program Test #3 (1)

- In-class Test: Nov. 12 (Tue.) 9:10-12:00 a.m.
- TA: 張維峻(d0657491@mail.fcu.edu.tw)
  林品秀 (d0676600@o365.fcu.edu.tw)
- Software: Dev-C++
- Submission:
  - Filename format:
    學號_PT#.c
    例如: M06455505_PT3.c
  - 12:00 a.m. 前，現場完成 demo，並且簽名
  - iLearn 2.0 submission (deadline: (Nov. 5 Tue. 23:59 p.m.)
    * 用自己的學號建立資料夾，並將 source code 放入資料夾，壓縮上傳 iLearn 2.0

Program Test #3 (2)

- Grading:

– Correctness	50%
– Program structure	20%
– Comments	10%
– Header block	5%
– Variable dictionary	10%
– Procedures and functions	5%

- Special notice:
  - 請勿抄襲別人程式(助教會當場進行測問、判定)，或是遲交作業，否則一律 0分計算
  - 請一律使用 C 語言來撰寫程式，且必須保證你的程式能夠再 Dev-C++ 軟體上成功編譯與執行，使用其他程式語言一律不予計分
  - 請依照題目給的輸入格式，否則不計分

Program Test #3 (3)

- Write a recursive C function to compute the longest and shortest paths of the tree.

Description:

有一棵時鐘樹，它為二元樹。樹中每個節點ni 都有一個延遲時間delay(ni)。對於一個由根到一個特定節點的路徑延遲長度定義為該路徑經過所有節點的延遲時間總和。寫出一個的遞迴 skew(treenode *root, int *longest, int *shortest) 函數來計算該樹的最長路徑跟最短路徑。其中skew函數會走過所有由根到樹葉節點的路徑，從中找到最長的路徑跟最短的路徑。

【注意】最長路徑指的是該路徑經過的所有節點延遲時間為最大的一條路徑。

Tree node defined as:

```
typedef struct listnode {
    int delay;
    struct node *lchild;        /* Pointer to the left substree */
    struct node *rchild;        /* Pointer to the right substree */
}treenode;
```

Basic requirements:

輸入: 給定Inorder(中序)跟Postorder(後序)的延遲時間數列，並根據上述節點結構建立該二元樹

輸出: 在螢幕上顯示該樹的最長路徑以及最短路徑的路徑延遲時間總和

Example:

(1) Input:

```
%> 13 18 30 51 5 22 7 2 42
%> 13 18 51 7 22 5 42 2 30
```

Output:

```
%> Delayed time (longest path) = 88
%> Delayed time (shortest path) = 61
```

(2) Input:

```
%> 21 33 41 59 5 32 29 14 43
%> 21 33 59 29 32 5 43 14 41
```

Output:

```
%> Delayed time (longest path) = 121
%> Delayed time (shortest path) = 95
```

Program Test #3 (4)

- Pseudo code: TOP DOWN

```
------------------------------------------------------------------------------------------------------------
void skew(treenode *root, int *longest, int *shortest)
{
  static int accumulative_delay = 0;
  accumulative_delay += root->delay;
  if(root->lchild == NULL && root->rchild == NULL)
  {
          if(*longest < accumulative_delay)
                  *longest = accumulative_delay;
          if(*shortest > accumulative_delay)
                  *shortest = accumulative_delay;
  }
  else
  {
          if(root->lchild != NULL)
                  skew(root->lchild, longest, shortest);
          if(root->rchild != NULL)
                  skew(root->rchild, longest, shortest);
  }
  accumulative_delay -= root->delay;
  return;
}
```

Program Test #3 (5)

- Pseudo code: BOTTOM UP

```
------------------------------------------------------------------------------------------------------------------------
void skew(treenode *root, int *longest, int *shortest)
{
  int llong, lshort, rlong, rshort;
  if (root == NULL) {
          *longest = 0;
          *shortest = 0;
          return;
  }

  skew (root->lchild, &llong, &lshort);
  skew (root->rchild, &rlong, &rshort);

  if ( llong >= rlong )
          *longest = llong;
  else
          *longest = rlong;
  if ( lshort < rshort )
          *shortest = lshort;
  else
          *shortest = rshort;

  *longest = *longest + root->delay;
  *shortest = *shortest + root->delay;
}
```