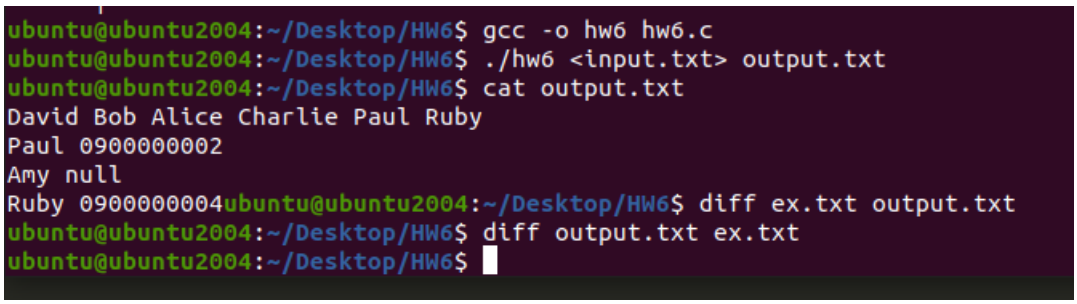
HW6

Result screenshot



Function

1. 內建函式:

```
char *strcpy(char *dest, const char *src)
```

這是用來將我讀入的資料直接複製到 struct 之中的 node 之中的 name 字元陣列或是 phonenumber 陣列裡面的函式。

```
int strcmp(const char *str1, const char *str2)
```

這是用來比對剛插入之節點與原本樹內節點之字典順序大小的函式。

自訂函式:

at create_AVL(char*,char*);

此為建立新的欲插入之節點的函式

at insert_AVL(at,at);

此為將新建好之節點插入 tree 內之函式

at search_AVL(at,char*);

此為用來搜尋目標節點的函式

int calculate_height(at);

此為用來計算左右子樹高度差之函式

void compare_name(const char*,const char*);

此為用來做字典順序比對之函式

at LL(at);

at LR(at);

at RL(at);

at RR(at);

這四個就是用來處理四種不同旋轉方式的函式

void preorder(at);
這就是 VLR 走訪函式

程式運行邏輯與我的思路:

這題比較特別的地方，第一個就是他和上次的二分搜尋樹不同，他是依照名字的字典排序順序去比較大小的，我本來想要一個字母一個字母比對，但是後來上網查到可以使用 strcmp 直接比對就好了。接著是正常的建立節點，在插入節點時要做的事很多，首先要先知道該節點要插在哪裡，這裡我有使用一個 judge 變數去判斷新插入的節點與原本節點之比較結果，接著就是去計算左右子樹的高度差，若有>1 的情況則必須判別是那四種旋轉情況的哪一種，並使用其中一種副程式去做旋轉。接著就是搜尋，印出，preorder，這邊比較沒有什麼特別的技巧，有許多都可以直接套用上次 hw4 的內容 code 來使用。值得一提的是，我的 main 函式裡面有許多額外的處理，都是為了讓自己想的測資能過，比如 DS 之間無數據，或是數據只有一個字元的情況。