參考資料：
http://www.sanfoundry.com/cpp-program-implement-binary-tree-2/

http://blog.csdn.net/here1009/article/details/8071251
心得：

Binary search tree 由 Insert(包含 search)、Inorder、Postorder，3 個 function 去組成，讀檔的部分很麻煩，因為和以前不同，這次還有：和、，所以不能用 getline 直接讀取整行，要分開，遇到空格、符號時要分別讀取做判斷

至於霍夫曼編碼課本寫太少，雖然知道往左子樹:0，往右子樹:1 的概念，我還是不知道如何下手，便查詢超多資料，大部分都是不能用的，不是 struct node 中的 leftchild->data 在進行運算時超級複雜，幾乎接近 300 行的 function 完全不能解讀，最後與同學討論找到這篇

```
1.  #include<iostream>
2.   #include<string>
3.   #include<queue>
4.   using namespace std;
5.
6.   class node{
7.       public:
8.          node(string con, float wht, node* left, node* right, string co
    ){
9.              content=con;
10.             weight=wht;
11.             leftchild=left;
12.             rightchild=right;
13.             code=co;
14.             }
15.         string content;
16.         float weight;
17.         node* leftchild;
18.         node* rightchild;
19.         string code;
20.      };
21.
22. void insertion_sort(node** array, int low, int high){
23.     for(int i=low+1;i<high;i++){
24.         node* tem=array[i];
25.         int j=i-1;
```

```
        while(array[j]->weight>tem->weight&&j>=low){
            array[j+1]=array[j];
            j--;
            }
        array[j+1]=tem;
        }
    }
void create_huffman_tree(string* s, float* w,int n,node** array){
    for(int i=0;i<n;i++){
        array[i]=new node(s[i],w[i],NULL,NULL,"");
        }
    insertion_sort(array,0,n);
    //~ for(int i=0;i<n;i++){
        //~ cout<<array[i]->content<<"*";
        //~ }
    int p=0;
    while(p!=n-1){
        node* min_1=array[p];
        node* min_2=array[p+1];
        node* new_node=new node("",min_1->weight+min_2->weight,min_1,min_2,"");
        //cout<<new_node->weight<<endl;
        array[p+1]=new_node;
        p=p+1;
        insertion_sort(array,p,n);
        }

    }

void create_huffman_code(node* p){
    queue<node*> nq;
    nq.push(p);
    while(nq.size()!=0){
        node* cur=nq.front();
        nq.pop();
        node* l=cur->leftchild;
        if(l!=NULL){l->code=cur->code+"0"; nq.push(l);}
        node* r=cur->rightchild;
```

```
63.         if(r!=NULL){r->code=cur->code+"1"; nq.push(r);}
64.         if(l==NULL&&r==NULL){
65.             cout<<cur->content<<": "<<cur->code<<endl;
66.             }
67.         }
68.     }
69. int main(int argc, char** argv){
70.     node* array[8];
71.     string s[8]={"a","b","c","d","e","f","g","h"};
72.     float w[8]={1,1,2,3,5,8,13,21};
73.     create_huffman_tree(s,w,8,array);
74.     create_huffman_code(array[7]);
75.     }
```

程式碼基本上將 main 中改為讀檔
但這還是失敗，output 無論如何都是空白
於是我便用同學的方法
將 Input 中的計算行數和行中的位置分開去讀
1 個 STR1， 1 個 STR2
避免數值在運算過程跑掉

```
fs.open("Input2.txt",fstream::in);                                    //open the file
        fp.open("Output2.txt",fstream::out|fstream::trunc);           //write the file
        int line = 0;
        while(getline(fs,str1)){
            line++;        //算幾行
        }
        /*************************************分開的目的是為了防止 2 邊的檔案互相混雜*************************************/
        /**********先算 Input 有幾行     再計算 Input 中的某行    他其中的字串*****/
        fs.close();
        fs.open("Input2.txt",fstream::in);
```