1.DecisionTree

1-1 Library:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<string.h>
```

1-2 Language: C
1-3 Environment: compiler:LLVM9.0.0 (but can also work with gcc5.4.0)
1-4 Results:

Because I randomly separate the traning data, results will be silghtly different in each running. However, no matter how the training data is changed, the predicion and the recall of Iris-setosa are always 1.
Below is one of my test result:

```
0.960
1.000 1.000
0.969 0.912
0.906 0.935
```

1-5 Usage:
程式中使用的函式與功能如下：

traversal—接收Testdata和root的位置，沿著root往下走，在每個node進行比大小，直到走至leaf node，就回傳預測結果。
DTree—建立決策樹的涵式，建立樹根(root)並呼叫insert_node。
insert_node—用遞迴的方式，不斷的製造左子樹與右子樹，終止條件為自身的rem=0。
calentropy—找出最大ig的函式。會依照儲存於threshold表中threshold值，計算這個threshold下的rem，找出ig最大的threshold。
parentropy—計算rem的函式。被calentropy呼叫。
creat_newdata—為新的node建立陣列。
totalenropy—計算entropy的函式。
threshold—找出所有的threshold並將他們建成表格。
randseq—打亂所有data。

2.Random Forest

2-1~2-3 同上
2-4 Results:
實際觀察下來，RF的預測結果其實不比DT好多少，有時甚至更差。我猜想或許是因為樣本數目過少的關係。

```
0.960
1.000 1.000
0.918 0.969
0.956 0.877
```

2-5 Usage:

Num of Tree:**15**

我的RF做法與DT雷同，基本上就是將一棵DTree延伸至好幾顆。一樣使用k-fold，只是training data的取樣方法不同。在DTree裡，all data扣掉test data後即為training data，RF則是將DT的trainging data 隨機取樣120次，取過的樣本仍可再取，結果則是RF的training data。程式碼中，在跑k-fold迴圈之內再新增一迴圈，圈數為樹的數目，每跑一次都會 重新取樣－建樹－回傳樹根－用陣列儲存所有樹的樹根位址。等所有樹都跑完後就用test data 依著不同的樹根進行traversal，統計最終結果，再計算recall、precision、accuarcy 等值。