資料結構作業10 書面報告

DEMO: https://youtu.be/4ianzBUBhlk

Tree * Stack

```
int tree_stack_full(Tree_Stack *s)
```

判斷存Tree * 的stack 是否存滿了

```
int tree_stack_empty(Tree_Stack *s)
```

判斷存Tree * 的stack 是否是空的

```
void tree_push(Tree_Stack *s, Tree *c)
```

將 Tree * c 存入 tree_stack 中

```
Tree *tree_pop(Tree_Stack *s)
```

從 tree_stack 中 pop out Tree*

Char Stack

```
int stack_full(Stack *s)
```

判斷存 Char 的 stack 是否存滿了

```
int stack_empty(Stack *s)
```

判斷存 Char 的 stack 是否是空的

```
void push(Stack *s, char c)
```

將 Char c 存入 stack 中

```
char pop(Stack *s)
```

從 stack 中 pop out Char

Tree Queue

```
void addq(Tree *queue[], int *front, int *rear, Tree *node)
```

將 Tree * node 存入 circular queue

```
Tree *deleteq(Tree *queue[], int *front, int *rear)
```

將 Tree * node 刪除於 circular queue

Infix to Postfix

```
char *infix_to_postfix(char *infix)
```

利用 Stack 將 infix 轉為 postfix ，並回傳 postfix

```
int isp(char c)
```

Stack 中 運算子優先值

```
int icp(char c)
```

準備放入 Stack 的運算子優先值

Postfix to binary tree

```
Tree *make_tree(char *postfix)
```

根據傳入 postfix，利用 tree stack 建置 binary tree，並回傳 root

Preorder traversal

```
void preorder(Tree *root)
```

利用遞迴，Preorder 走訪 binary tree

Level-order traversal

```
void level_order(Tree *root)
```

利用 tree queue，Level-order 走訪 binary tree

Evaluates the expression

```
void calculate(Tree *root)
```

利用 postorder traversal 走訪 binary tree ，並計算數值

Validates the Infix expression

```
char operand(char c)
```

根據傳入的 char c，分成 運算子（o）、左括弧（l）、右括弧（r）和大寫字母（a），並回傳代表值

```
int valid(char *infix)
```

判斷 Infix 是否 valid