Lab7 Tree Build, Traverse and Expression

- 검사절차:

Input: 2+4*3 ➔ Output: (2 + (4* 3))

1. Node Creation:

```
class node { public:
        Char    data;     // one character input per node ex) A
        Int       prio;     // priority number from precedence table
        node *left;    // left    link
        node *right;    // right    link
  }
```

2. Precedence Table

```
char prec[4][2] = { '*', 2,    '/', 2,    '+', 1,    '-', 1};
```

=>

*	/	+	-
2	2	1	1

3. Main Program

1) Get math expression in numbers (ex: 2+4*3)

2) Build Tree

3) Traverse tree (Inorder, Preorder, Postorder)

4) Tree Expression using Parentheses

* Details

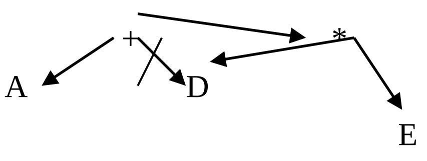
1) Get math expression(수식 입력): 키보드 에서 입력.

2) Build Tree

```
while (input !=NULL)
  { . create new-node
   . assign DATA-INPUT into new-node's data field & default prio '4'
   . for i=0 to 4    (if new-node-> data == prec[i][0])
                          then new-node->prio = prec[i][1]
   . if (i==5)    then    call Operand(new-node)
                  else    call operator(new-node)
 } endwhile
```

```
* Operand(new-node)
    If HEAD==NULL   then    HEAD=new-node    return
    P = Head
    While (p->right !=NULL)    p=p->right
    P->right = new-node

* Operator (new-node)
     if (head->prio   >=   new-node->prio)
          new-node->left = Head
          Head = new-node
     Else
          New-node->left = Head->right
          Head -> right = new-node
```



3) Traverse (Tree traverse algorithm 참조): Inorder, Preorder, Postorder

4) Tree Expression using Parentheses

```
Procedure parentheses (node) {
    if (node != NULL) {

        if (!isdigit(node->data))        cout << " ( ";

          parentheses(node->left);              //infix 형태의 출력
          print node->data;
          parentheses(node->right);

        if (!isdigit(node->data))        cout << " )";
    }
}
```