

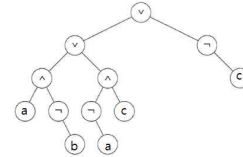
The first screenshot shows the initial state of the algorithm. The command prompt title is "C:\WINDOWS\system32\cmd.exe". The input is the infix expression $a + b * c + d$. The output shows the infix expression as $a|b * c|+d|$ and the postfix expression as $a b * c + d|$. The Korean text "계속하려면 아무 키나 누르십시오 . . ." is visible at the bottom.

The second screenshot shows the next step in the process. The command prompt title is the same. The input is the infix expression $a | (b * c) | a * b$. The output shows the infix expression as $a|(b * c)|a * b|$ and the postfix expression as $a b c * + a b *|$. The Korean text "계속하려면 아무 키나 누르십시오 . . ." is visible at the bottom.

2. [충족 가능 문제(Satisfiability Problem)] postfix expression의 명제식을 파일로부터 입력 받아 이진트리를 구성하여 중위순회(inorder traversal)한 결과를 화면에 출력 한 후, 충족가능문제의 모든 조합에 대한 결과를 화면에 출력하는 프로그램을 작성하라. (10점)

[입력파일 및 이진트리]

- 입력파일(postfix.txt) : ab~&a~c&|c~|
- 피연산자(Operands) : 알파벳 소문자
- 연산자(Operators) : & | ~



[실행 순서]

- ① postfix 명제식 (1번 문제의 postfix.txt 활용 가능)으로 부터 이진트리를 생성한다.
※ 자료구조응용15. 1번 소스를 수정하되, 트리 노드 정의는 구현 세부 사항을 참고
- ② 이진트리에 대한 중위 순회를 수행하여 명제식을 출력한다.
- ③ 명제식의 변수 (a, b, c)에 대해 가능한 모든 값의 조합에 대해 진리값을 출력한다.

[구현 세부 사항]

- ① 충족가능문제를 풀기 위한 명제식의 트리정의

```
// satisfiability problem
#define TRUE 1
#define FALSE 0
typedef enum{ NOT, AND, OR, true, false} logical;

typedef struct node *treePointer;
typedef struct node
{
    treePointer leftChild;
    char var;           // propositional variables a, b, c
    logical data;       // [ NOT, AND, OR for operator ] [ true, false for variable ]
    short int value;    // TRUE, FALSE
    treePointer rightChild;
}tNode;

treePointer root = NULL;
logical aData, bData, cData; // combination for a, b, c - ex) (true, false, true)
void postOrderEval(treePointer node); // Program 5.9. postorder evaluation function
...
```

- ② 명제식의 이진트리 생성

```
// binary tree of an propositional expression
char expr[81]; // postfix expression
typedef enum { lparen, rparen, or, and, not, eos, operand } precedence;
precedence getToken(char *symbol, int *n);
treePointer createBinTree(void);

// binary tree traversals
void inorder(treePointer ptr);
```

※ 주의

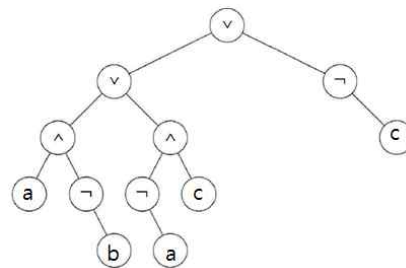
- 교재 소스에서 TRUE, FALSE는 기호상수로 정의된 것이다.
- 소스파일 확장자를 .c로 하라!** .cpp로 하는 경우, Visual C++은 C++ 컴파일러로 컴파일하므로 bool 타입 예약어인 true, false 때문에 교재 소스를 그대로 사용할 수 없게 된다.
- logical** 과 **precedence** 열거형에서 **logical and**, **logical or**, **logical not**을 표현하는 상수를 각각 대소문자를 달리하여 표현하였다. (기존 소스 수정을 최소화하여 활용하기 위해 서임. 이 부분도 각자의 생각대로 효율적인 방법으로 구현하면 됨)
- inorder()** 함수는 실행예와 같이 출력되도록 수정한다.
- 위 소스는 단순히 참고 사항일 뿐**, 각자 자기 스타일과 알고리즘대로 구현하면 됩니다!!
- 학생들의 창의적인 프로그래밍을 위해 기타 함수 설명은 생략합니다!

[실행예]

```

C:\WINDOWS\system32\cmd.exe
***** satisfiability problem *****
Note: For efficiency, &, |, ~ are used instead of
logical and(&&), logical or(||), logical not (!)
*****
input string(postfix) : ab~&a~c~|c~|
creating binary tree for satisfiability algorithm...
inorder traversal      : a&b|~a&c|~c      7점
for all combinations of (a, b, c)...      3점
(t, t, t) is FALSE
(t, t, f) is TRUE
(t, f, t) is TRUE
(t, f, f) is TRUE
(f, t, t) is TRUE
(f, t, f) is TRUE
(f, f, t) is TRUE
(f, f, f) is TRUE
계속하려면 아무 키나 누르십시오 . . .

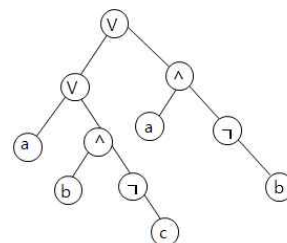
```



```

C:\WINDOWS\system32\cmd.exe
***** satisfiability problem *****
Note: For efficiency, &, |, ~ are used instead of
logical and(&&), logical or(||), logical not (!)
*****
input string(postfix) : abc~&|ab~&|
creating binary tree for satisfiability algorithm...
inorder traversal      : a|b~&c|a~&b
for all combinations of (a, b, c)...
(t, t, t) is TRUE
(t, t, f) is TRUE
(t, f, t) is TRUE
(t, f, f) is TRUE
(f, t, t) is FALSE
(f, t, f) is TRUE
(f, f, t) is FALSE
(f, f, f) is FALSE
계속하려면 아무 키나 누르십시오 . . .

```



■ 제출 형식

- 솔루션 이름 : DS 16
- 프로젝트 이름 : 1, 2
- 소스파일 이름 : 1.c, 2.c
- 각 소스 파일에 주석처리
“학번 이름”

“본인은 이 소스 파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”

- 제출 파일

- ① 소스코드와 실행 결과가 보이도록 화면을 캡처한 **보고서** 파일(“학번.pdf”)

※ 한글 [파일 → pdf로 저장하기...] 메뉴 사용

- ② **C 소스 파일**을 하나의 디렉터리에 모아 압축한 파일 (“학번.zip”)

※ “학번.pdf”와 “학번.zip”을 하나로 압축하지 말고 별도 파일로 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감 : 수업일 자정
- 2차 마감 : 수업 익일 자정(만점의 80%, 반올림) 이번 문제만 80%로 조정함
- 문항 별로 1차 2차 나눠서 제출할 수 없으며, 최종 제출 시간에 따라 1차, 2차로 구분함