CSE 3302 Lab Assignment 3

Due October 26, 2012

Goal:

Understanding of boolean expressions, short-circuit evaluation, and branching.

Requirements:

1. Design, code, and test a program to translate a postfix boolean expression into a branching program whose execution will be simulated. The input is a single string with the following properties:

   a. No more than 100 symbols from the set { `T`, `F`, `D`, `&`, `|`, `!` }. No whitespace, no parentheses, etc.
   b. The distinction between the symbols `T`, `F`, `D` is only relevant during execution of the branching program. `D` means "die" and stops the simulation. During translation these symbols correspond to boolean operands and should be numbered starting at `0` going from left-to-right through the input. There will be no more than 30 operands in an expression.

   The required outputs are:

   a. The echoed input string.
   b. The branching program as (index, operation, destination) triples. Your program will have as many lines as there are operands in the input string, with each index/line corresponding to one of the operands. An operation can be either `brT` (branch on true) or `brF` (branch on false). The destination will be the index of another triple later in the program. During execution, the program counter will continue at the destination if the condition is satisfied. Otherwise, the program counter is incremented to the next triple. If the result of an entire input expression is true, the program counter should be set to the number of operands. Otherwise, it should be set to number of operands + 1.
   c. An execution trace listing the indices of the triples involved in the execution.

2. Email your program to `mehra.nourozborazjany@mavs.uta.edu` by 12:45 p.m. on October 26.

Getting Started:

1. Some example outputs:

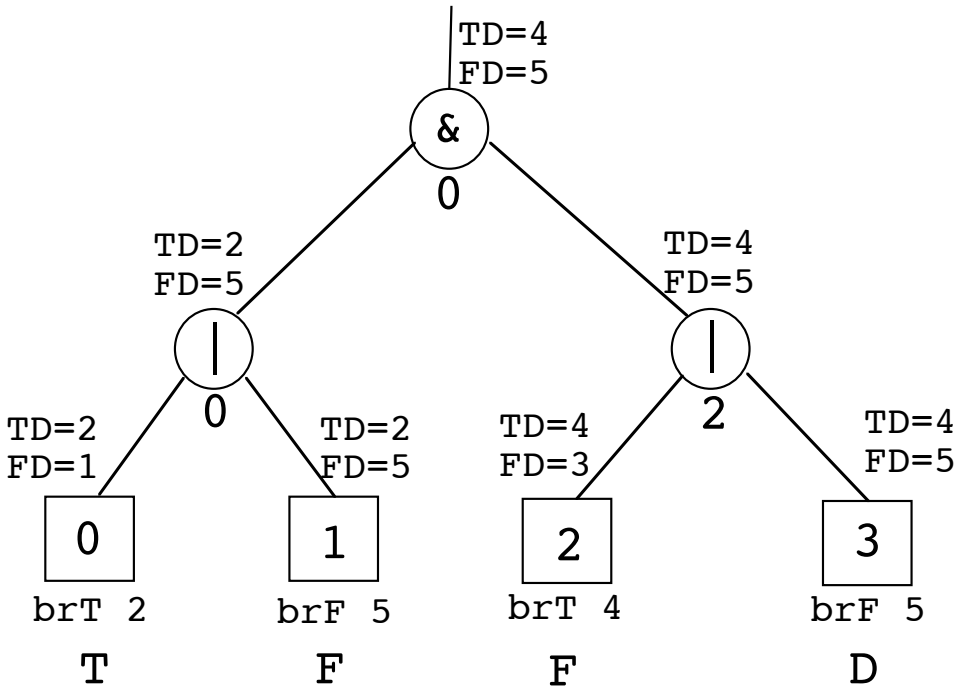
```
TF|
Method 1 generated branch program:
0 brT 2 (T)
1 brF 3 (F)
Evaluating:
At state 0
Evaluates true

FF|!TD|&!
Method 1 generated branch program:
0 brT 4 (F)
1 brT 4 (F)
2 brT 5 (T)
3 brT 5 (D)
Evaluating:
At state 0
At state 1
At state 2
Evaluates false
```

```
TF|FD|&
Method 1 generated branch program:
0 brT 2 (T)
1 brF 5 (F)
2 brT 4 (F)
3 brF 5 (D)
Evaluating:
At state 0
At state 2
At state 3
Die
```

2. This assignment may be done using an expression tree:

   a. Read input and use a postfix evaluator to construct tree.
   b. Propagate the smallest operand number (i.e. leftmost leaf) to the root of each subtree. Essentially, these are synthesized attributes.
   c. Propagate downward through the tree (i.e. like inherited attributes), the destinations for branching when the result of a subexpression becomes known (for both true and false). At the root the true destination is the number of operands, while the false destination is the number of operands + 1. These values are elegantly manipulated during the propagation and allow setting the branch destinations in the leaves.
   d. Determining whether the instruction is a `brT` or `brF` depends on the fall-through property of the program counter. You will never have a triple of form (`i`, `brX`, `i+1`) and you will need a simple correction strategy when this occurs.



3. It is possible to implement the translation in one pass without the tree (by simulating postfix evaluation). It involves bringing the unresolved instruction locations (as sets) to the corresponding destinations. Unfortunately, it is very helpful to implement the simpler approach first.