Functional and logic programming lab 2nd report

Yoshiki Fujiwara, 05-191023

1 Q.1: Peano Arithmetic

1.1 An example operation

Listing 1 test code

```
add (S (S Z)) (S Z);; -> nat = S (S (S Z))

sub (S (S Z)) (S Z);; -> nat = S Z

sub (S Z) (S (S Z));; -> nat = Z

mul (S (S Z)) (S (S (S Z))) nat = S (S (S (S (S Z)))))

pow (S (S Z)) (S (S (S Z)));; -> nat = S (S (S (S (S (S (S Z)))))))

n2i (S (S Z));; -> int = 2

n2i Z;; -> int = 0

i2n 2;; -> nat = S (S Z)
```

1.2 Discussion

This code is about peano arithmetic. Z means zero and S means successor function. Adding n is written as applying successor function n times. Subtructing n is written as removing S n times. Multiplication is written using additional function. Also power function is written using multiplication.

2 Q.2: DFS

2.1 An example operation

Listing 2 test code

```
pre_order(Node('a', Node('b', Node('d', Leaf, Leaf),
 Node('e', Leaf, Leaf)), Node('c', Leaf, Node('f',
 Node('g', Leaf, Leaf), Leaf))));;
 -> char list = ['a'; 'b'; 'd'; 'e'; 'c'; 'f'; 'g']

in_order(Node('a', Node('b', Node('d', Leaf, Leaf),
 Node('e', Leaf, Leaf)), Node('c', Leaf, Node('f',
 Node('g', Leaf, Leaf), Leaf))));;
 -> char list = ['d'; 'b'; 'e'; 'a'; 'c'; 'g'; 'f']

level_order (Node('a', Node('b', Node('d', Leaf, Leaf),
 Node('e', Leaf, Leaf)), Node('c', Leaf, Node('f',
 Node('g', Leaf, Leaf), Leaf))));;
 -> char list = ['a'; 'b'; 'c'; 'd'; 'e'; 'f'; 'g']
```

These test codes are applied to a tree like the below pictures

2.2 Discussion

2.2.1 About code

The written code is very simple. First, divide type 'a tree into three pieces and then concat them in the specific order. That is all I did.

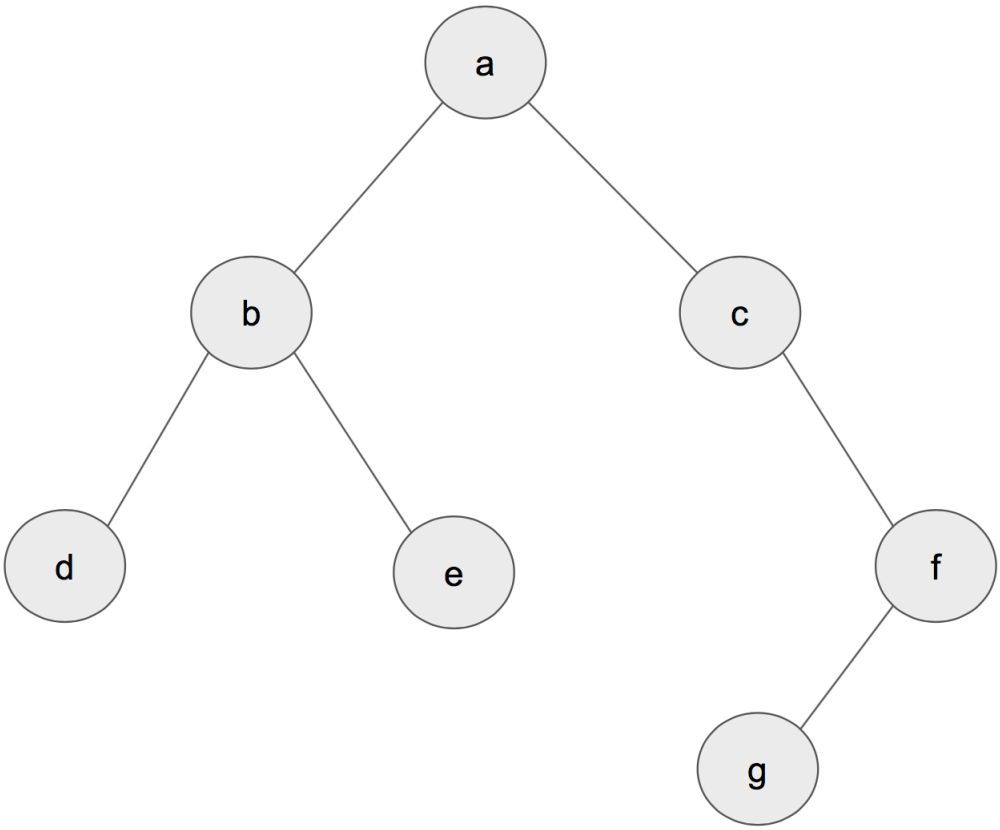
2.2.2 Application

Pre-order outputs the node root as the first, left as the second and right as the last. Pre-order traversal is used to used to get a prefix expression of an expression tree.

In-order outputs the node left as the first, root as the second and right as the last. In-order traversal is used in case of binary search tree. In-order traversal gives us the nodes in BST n non-decreasing order.

Post-order outputs the node left as the first, right as the second, and root as the

last. Post-order traversal is used to delete the tree.



3 Q.3: BFS

3.1 An example of operation

Listing 3 test code

```
level_order (Node('a', Node('b', Node('d', Leaf, Leaf),
 Node('e', Leaf, Leaf)), Node('c', Leaf, Node('f',
 Node('g', Leaf, Leaf), Leaf))));;
-> char list = ['a'; 'b'; 'c'; 'd'; 'e'; 'f'; 'g']
```

This example tree is the same as the above figure.

3.2 Discussion

3.2.1 About code

In the code, I created a list that stores type 'a tree. Recursion is operated to the list as follows.

1. Take an element 'a tree, from the list.
2. If the element is a node, then memorizes the node in the accumulator.
3. Then, push two subtrees (left and right) into the bottom of the list.
4. Run 1. - 3. again and again until a node becomes Leaf.

3.2.2 Application and comparison with Q.2

BFS is used in many ways, for example to find the shortest path and to compute the maximum flow. DFS is easier to write than BFS, because there is no need to make a list that stores subtrees. DFS is more compatible with funtional programming.

4 Q.4 and Q.5

Listing 4 test code