CEng 242 Homework 1

Due: $2^n d$ March 2001

Assume you are given the following datatype definition for a tree:

```
datatype inttree = Empty | Node of int*inttree*inttree;
```

Note that a leaf is a `Node` with `Empty` leaves (e.g. `Node(2,Empty,Empty)`).
A function called diffsum*:* `inttree`$\times$`inttree` $\rightarrow$ `int` is defined on two trees as follows:

$$\mathsf{diffsum}(A,B) = |val(A) - val(B)| + \mathsf{diffsum}(left(A), left(B)) + \mathsf{diffsum}(right(A), right(B))$$

Where $val(A)$ is the integer in the root node of A, $left(A)$ is the left subtree of A and $right(A)$ is the right subtree of A, and:

$$\mathsf{diffsum}(A, \texttt{Empty}) = \mathsf{diffsum}(\texttt{Empty}, A) = \text{sum of absolute values from all nodes of } A$$

Another function mindiffsum*:* `inttree`$\times$`inttree` $\rightarrow$ `int`$\times$`inttree` is defined as:

> $\mathsf{mindiffsum}(A,B) = (\mathsf{diffsum}(A', B), A')$
> where A' is the subtree of A with minimum $\mathsf{diffsum}(A', B)$ value,
> i.e. $\forall A^*, A^*$is a subtree of $A, \mathsf{diffsum}(A', B) \leq \mathsf{diffsum}(A^*, B)$

In case of equality, $\mathsf{diffsum}(A', B) = \mathsf{diffsum}(A'', B)$ for any two subtrees, mindiffsumshould return the tree:

- if one of the trees is the antecedent of the other, the antecedent is returned
- if there is no antecedent relation among two trees, the leftmost subtree is returned

Write an ML program to implement diffsum and mindiffsum functions defined above. Functions should have the following ML type signatures:

```
  diffsum:  inttree*inttree -> int
  mindiffsum:  inttree*inttree -> int*inttree
```

Examples:

```
val a=Node(3,Node(5,Empty,Empty),
             Node(~1,Node(7,Empty, Empty),
                     Node(9,Empty,Empty)));

val b=Node(~1,Node(2,Node(3,Empty,Empty),
                     Node(7,Empty,Empty)),
              Node(1,Empty,Empty));

printtree(a);
             /9
      / ~1 -|
     |       \7
+ 3 -|
      \5

printtree(b);
```

```
       /1
+ ~1 -|
      |      /7
       \ 2 -|
             \3

diffsum(a,b);
val it = 35 : int
```

diffsum(a,b)=|3-(~1)| + |5-2| + |~1-1| + 7 + 9 + 3 + 7 = 35

```
val c= Node(7,Node(~3,Node(2,Empty,Empty),
                      Node(14,Empty,Empty)),
              Node(~3,Node(~2,Empty,Empty),
                      Node(10,Empty,Empty)));

val d= Node(3,Node(5,Node(~2,Empty,Empty),
                      Node(11,Empty,Empty)),
              Node(9,Node(0,Empty,Empty),
                     Node(7,Empty,Empty)));

printtree c;
             /10
      / ~3 -|
     |       \~2
+ 7 -|
     |       /14
      \ ~3 -|
             \2

printtree d;

            /7
      / 9 -|
     |      \0
+ 3 -|
     |      /11
      \ 5 -|
            \~2

val (ds,dtree)=mindiffsum(c,d);
val ds = 34 : int
val dtree = Node (~3,Node (2,Empty,Empty),Node (14,Empty,Empty)) : inttree

printtree dtree;
       /14
+ ~3 -|
       \2
```

diffsum(c,d)=36 , diffsum('left subtree of c', d)=34, diffsum('right subtree of c',d)=34 . All leaf nodes of c returns larger values. Left subtree is returned.

You can use the following `printtree` function to pretty print your trees:

```
datatype direction = Up | Down | None | All;

fun indent _ None = "      "
  | indent _ All  = "|      "
  | indent x y = if (x=y) then "       "
                          else "|      ";

fun symb  None = "+"
  | symb  Up = " /"
  | symb  Down = " \\"
  | symb All = "+-";

fun print1 c d Empty =  c ^ (symb d) ^ "*" ^ "\n"
  | print1 c d (Node(x,Empty,Empty)) = c ^ (symb d) ^ (Int.toString x) ^ "\n"
  | print1 c d (Node(x,st1,st2)) =
                (print1 ( c ^ (indent Up d)) Up  st2 ) ^
         c ^ (symb d) ^ " " ^ (Int.toString x) ^ " -|\n" ^
                (print1 (c ^ (indent Down d)) Down st1 ) ;

fun printtree t = print (print1 "" None t);
```

Don't forget that those who cheat in any homework will get 0 from all homeworks and punishment will be the same for both source and the cheater. More test data and submission details will be posted to `metu.ceng.course.242` later. Please send all questions to this newsgroup. Good luck.