CS 496 – Inductive Sets – Exercise Booklet 2

Exercise 1

Consider the following grammar

$$
\begin{array}{rcl}
\langle\mathsf{Number}\rangle & ::= & n, \text{ with } n \in \mathbb{N} \\
\langle\mathsf{E}\rangle & ::= & \langle\mathsf{Number}\rangle \\
 & | & \langle\mathsf{E}\rangle + \langle\mathsf{E}\rangle \\
 & | & \langle\mathsf{E}\rangle - \langle\mathsf{E}\rangle \\
 & | & (\langle\mathsf{E}\rangle)
\end{array}
$$

1. Identify the terminals and nonterminals.
2. Identify the productions.
3. Give a derivation showing that the following sequence of terminals belongs to the grammar: $1 + (4 - 7)$
4. Give three additional examples of sequences of terminals that form syntactically correct expressions

Exercise 2

Consider the following grammar for binary tree expressions.

$$
\begin{array}{rcl}
\langle\mathsf{SimpleBTree}\rangle & ::= & \langle\mathsf{Number}\rangle \\
 & | & \{*\langle\mathsf{SimpleBTree}\rangle\langle\mathsf{SimpleBTree}\rangle*\}
\end{array}
$$

Write a derivation to show that `{*2 {*3 4*}*}` is generated by the non-terminal <SimpleBTree>.

Exercise 3

Define the inductive sets given by the grammar of the first exercise in OCaml.

Exercise 4

Define the following simple inductive sets in OCaml:

- `Coordinate` that represents a coordinate in the plane and define the operations `getX`, `getY`, `distance` and `add`.
- `Shape` that represents either a circle or a rectangle in the plane. Define operations `area`, `perimeter` and `move`.

- `Shape3D` that represents either a cube, a cilinder or a sphere. Define operations `area` and `volumen`.
- `Person` that has a name, age, phone number and address.

Exercise 5

Consider the following inductive definition of binary tree expressions from exercise 2.

```
type SBTree = Leaf of int | Node of SBTree*SBTree
```

1. How is the tree `{*2 {*3 4*}*}` represented as an element of this inductive definition?
2. Write a function `is_node` of type `SBTree -> bool` that returns true if the argument is a tree whose root is an internal node.
3. Write a function `is_leaf` of type `SBTree -> bool` that returns true if the argument is a `SimpleBTree` that is a leaf.
4. Write a function `string_of_sbtree` of type `SBTree -> string` that converts a binary tree into a string. For example,

   ```
   > pretty_print t;;
   "(2 (3 4))"
   ```

   where `t` is defined to be the tree

   ```
   Node(Leaf 2,Node(Leaf 3,Leaf 4))
   ```

Exercise 6

Define three functions `preorder`, `inorder` and `postorder` that returns the standard traversals of a `BTree`. For example, if `t` is the tree:

```
let t= Node(6,
           Node(2,Leaf 1,
                     Node(4,Leaf 3, Leaf 5)),
          Node(7, Leaf 8, Leaf 9))
```

then

```
> inorder t;;
[1;2;3;4;5;6;8;7;9]
> preorder t;;
[6;2;1;4;3;5;7;8;9]
> postorder t;;
[1;3;5;4;2;8;9;7;6]
```

Exercise 7

Write a function btree_product that multiplies all the numbers in the tree. For example,

```
> btree_product (Leaf 8);;
8
> btree_product (Node(4,Leaf 1, Leaf 9));;
36
```

Exercise 8

Write a function btree_element that given a number and a BTree returns a boolean indicating whether the number belongs to the tree or not. For example,

```
> btree_element 8 ex;;
false
> btree_element 11 ex;;
true
```

Exercise 9

Write a function btree_bimap that given a two functions fLeaf and fNode and a BTree returns a new BTree resulting from the original one where fLeaf has been applied to the numbers in the leaves and fNode has been applied to the numbers in the nodes. For example,

```
> btree_bimap (fun x -> x+1) (fun y -> y+y) ex;;
Node(4,
     Node( 24, Leaf 8, Leaf 12),
     Node( 8,  Leaf 2, Leaf 10))
```

Note that in the new tree the leaves have been incremented by one but the numbers in the internal nodes have been doubled. Check this!

Exercise 10

Write a function btree_max that given a BTree returns the maximum number in the tree. For example,

```
> btree_max ex;;
12
```

Exercise 11

Write a function btree_bst that given a BTree returns a boolean indicating whether the tree is a binary search tree. For example,

```
> btree_bst ex;;
false
> btree-bst
        (Node( 12,
            (Node( 7, Leaf 2, Leaf 11)),
            (Node( 18, Leaf 15, Leaf 19))));;
true
```

Hint: you may assume that you have btree_min at your disposal (which returns the smallest number in a BTree).

Exercise 12

Write a function `btree_to_number` that given a `BTree` returns the number in its root, be it a leaf or an internal node. For example,

```
> btree_to_number (Leaf 8);;
8
> btree_to_number ex;;
2
```

Exercise 13

Write a function `level` returns a list of all the numbers at level `n` of a `BTree`. The first level of a tree is 0, the second is 1 and so on. If there are no nodes at level `n`, the empty list should be returned. For example,

```
> level 0 ex;;
[2]
> level 1 ex;;
[12;4]
> level 2 ex;;
[7;11;1;9]
> level 3 ex;;
[]
```