Modular Programming

Michael P. Fourman

October 29, 2006

Aims

In this practical, you will build a basic library of ML functors that should be useful in the project.

Assessment

Your work will be assessed on the basis of the correctness and efficiency of the functors you implement. You will not be expected to provide implementations more efficient than those already covered in the notes and examples (but you will be expected to use the best implementations we have introduced so far).
Your code should be placed in the directory `CS201/Prac5` under your home directory.

Deadline

The deadline for this practical is 6.00pm, Friday 22nd April.

1. You should implement functors (implementing the appropriate datatypes) with the following headers:

```
functor QUEUE(type Item):QueueSig
functor STACK(type Item):QueueSig
functor PQUEUE(type Item val >: Item * Item -> bool):QueueSig

functor EQSET(type Item val ==:Item*Item -> bool):SetSig
functor ORDSET(type Item val >:Item*Item -> bool):SetSig

functor ASSOCLIST(type Item eqtype Key):DictSig
```

The signatures referred to above are included in the `prac5` ML database (the identifier `==` is also declared to be infix). The signatures `SetSig` and `QueueSig` are given in the notes; `DictSig` is given below.

```
signature DictSig =
sig
   type Key
   type Item
   type  Dict
   exception Lookup
   val empty  :  Dict
   val lookup :  Dict -> Key -> Item
   val remove : Key *  Dict ->  Dict
   val enter  : (Key * Item) *  Dict -> Dict
end
```

The signature `DictSig` refers to a dictionary—akin to the `Environment` used in the example covered in Lecture Note 11, but that does not include a `remove` operation—in which you can enter, and remove pairs consisting of a key and an associated item, and lookup the item associated with a given key. Your functor should use a list of pairs to implement the signature.

The code for each functor should be placed in a file with the corresponding name, plus a `.ML` extension (for example, the code for the functor `QUEUE` should be placed in a file named `QUEUE.ML`).

2. You should also implement structures `IntItem`, `StringItem`, `RealItem` that can be passed as arguments to your functors, and use these to test your functors. Again, the code for each structure should be placed in a correspondingly named file with the `.ML` extension.