Principles of Programming, Spring 2006
Practice 9
Patterns in Function Definition and User-defined Data Types

Daejun Park, Heejong Lee
Programming Research Lab.@SNU

May 18, 2006

1. Define a procedure `reverse` that takes a list as argument and returns a list of the same elements in reverse order.

```
> val x = reverse [1, 2, 3, 4]
> ;;
val x: int list = [4, 3, 2, 1]
```

2. Consider the `mergeSort` procedure in the following. Define the sub procedures, `split` and `merge`, in order to run `mergeSort` well.

```
fun mergeSort [] = []
  | mergeSort [a] = [a]
  | mergeSort l =
      let
        val (m, n) = split l
        val m' = mergeSort m
        val n' = mergeSort n
      in
        merge m' n'
      end
```

3. The `diff` procedure takes a polynomial as its argument and differentiates the given polynomial. Fill in the missing expressions in the following definition of `diff`.

```
type poly = Add of poly * poly
          | Term of coef * expo
 and coef = int
 and expo = int
```

```
val rec diff : poly -> poly =
 fn Add (p, q) => <??>
  | Term (c, 0) => <??>
  | Term (c, e) => <??>
```

If `diff` has been completed correctly, it will have a result such as the following.

```
> val z = diff Add (Add (Term (1, 3), Term (4, 1)), Term (-5, 0))
> ;;
val z: poly = Add (Add (Term (3, 2), Term (4, 0)), Term (0, 0))
```