CS 3721: Programming Languages Lab

Lab #10: Tail Recursion and Loops

Suppose we have the following function definition *length* in ML.

```
fun length(nil, result) = result
   |  length(x::y, result) = length(y, result+1);
```

You can test the above function with the following invocation

```
- length ([2,3,4,5,7],0);
val it = 5 : int
```

The above *length* can be translated to the following loop implementation.

```
fun length(y) = let val result=ref 0; val p_y=ref y  in
                              while not (!p_y = []) do
                                  ( result := !result + 1; p_y := tl(!p_y) );
                              !result
                            end;
```

Translate the following tail-recursive ML functions to loop implementations. Test your code by invoking both the original recursive implementation and your new implementation with a test input, and make sure they return the same result.

1.
   ```
   fun Append(nil, ys) = ys
     |     Append(x::xs, ys) = Append(xs, x::ys);
   ```

2.
   ```
   fun Product([], res) = res
      | Product(x::y, res) = Product(y, x * res);
   ```

3.
   ```
   fun gcd(x,y) = if x = y then x
             else if (x > y) then gcd(y, x-y) else gcd(x, y-x);
   ```