Aufgabe 7 Paralleles Map + Reduce [25 Punkte]

In dieser Aufgabe soll eine Funktion `val map_reduce : ('a -> 'b) -> ('b -> 'b -> 'b) -> 'a list -> 'b` parallel mittels Threads implementiert werden. Die Funktion bildet dabei zunächst die Elemente einer nicht leeren[1] Liste mittels einer Funktion `f` ab (*map*-Phase). Über die Ergebnisliste wird anschließend eine zweite Funktion `g` gefaltet (*reduce*-Phase). Insgesamt kann `map_reduce` unter Nutzung nur eines Threads also wie folgt implementiert werden:

```
1 let map_reduce f g l =
2   let l' = List.map f l in
3   List.fold_left g (List.hd l) (List.tl l')
```

Um diese Funktion effizient parallelisieren zu können, setzen wir zusätzlich voraus, dass `g` assoziativ und kommutativ ist; typische Beispiele für derartige Funktionen sind die Addition und die Multiplikation.

Gehen Sie entsprechend der folgenden Teilaufgaben vor.

1. Implementieren Sie die Funktion `val pmap : ('a -> 'b) -> 'a list -> 'b event list`, die parallel eine Funktion `f` auf jedes Element einer Liste anwendet. Jede Anwendung von `f` soll dabei in einem eigenen Thread geschehen. Das Ergebnis ist eine Liste von Ereignissen. Jedes Ereignis entspricht dem Ergebnis der Anwendung von `f` auf das jeweilige Listenelement. **Hinweis**: `pmap` soll nicht auf die Fertigstellung der asynchronen Berechnung warten!

2. Implementieren Sie die Funktion `val preduce : ('a -> 'a -> 'c) -> 'a -> 'a -> 'c event`, die zwei Elemente mittels einer Funktion `g` parallel vereint. Die Vereinigung soll dabei in einem eigenen Thread geschehen. Das Ergebnis ist ein Ereignis, dessen Eintreten der Fertigstellung der Anwendung von `g` entspricht. **Hinweis**: `preduce` soll nicht auf die Fertigstellung der asynchronen Berechnung warten!

3. Implementieren Sie die Funktion `val reduce_list : ('a -> 'a -> 'a) -> 'a event list -> 'a`, die als Parameter eine Funktion `g` sowie eine Liste von Ereignissen (Rückgabe von `pmap`) erwartet. Die Funktion wartet wiederholt auf das Eintreten zweier beliebiger Ereignisse aus der Ereignisliste. Die beiden Ereignisse liefern zwei Werte, aus denen mittels `preduce` ein neues Ereignis erzeugt wird; das neue Ereignis wird dann in die Liste eingefügt. Die Menge der ausstehenden Ereignisse schrumpft so sukzessive, bis nur mehr ein Ereignis übrig bleibt, dessen Ergebnis als Rückgabewert der Funktion dient.

4. Implementieren Sie die Funktion `val map_reduce : ('a -> 'b) -> ('b -> 'b -> 'b) -> 'a list -> 'b` auf Basis obiger Funktionen.

[1] Sie müssen Fehler nicht behandeln.