```
functor MkSkyline(structure S : SEQUENCE) : SKYLINE =
struct
  structure Seq = S
  open Seq

  fun unzip s =
      (map (fn (a,_) => a) s, map (fn (_,b) => b) s)

  fun skyline buildings =
      let
        (* addXs toAdd sky ==> sky'
         *
         * addXs merges the x-coordinates from 'toAdd' into the
         * skyline described by sky, copying heights from sky
         * over each new x-coordinate to give a new skyline sky'.
         *)
        fun addXs toAdd sky =
            let
              fun copy ((_, SOME h), (x, NONE)) = (x, SOME h)
                | copy (_, r) = r
              val copyScan = scani copy (0, NONE)

              val newXs = map (fn x => (x, NONE)) toAdd
              val oldSky = map (fn (x,h) => (x, SOME h)) sky
              fun cmpX ((x1,_), (x2,_)) = Int.compare (x1,x2)
            in copyScan (merge cmpX newXs oldSky)
            end

        fun combine (sky1, sky2) =
            let
              fun optMax ((NONE, x) | (x, NONE)) = valOf x
                | optMax (SOME x, SOME y) = Int.max (x, y)

              val xsOf = map (fn (x,_) => x)
              val (xs, hs1') = unzip (addXs (xsOf sky2) sky1)
              val (_,  hs2') = unzip (addXs (xsOf sky1) sky2)
            in zip xs (map2 optMax hs1' hs2')
            end

        val init = map (fn (l,h,r) => %[(l, h), (r, 0)])
        val (xs, hs) = unzip (reduce combine (empty ()) (init buildings))

        fun isUniq (0, _) = true
          | isUniq (i, (x,h)) = nth hs (i-1) <> h

      in filterIdx isUniq (zip xs hs)
      end
end
```