```
(with-primitive-procedures [unary-potential binary-potential]
  (defquery square-lattice-ising [n a b]
    (let [unary (unary-potential a)
          pairwise (binary-potential b)]
      (loop [i 0
             rows []]
        (if (= i n)
          (predict :x rows)
          (let [next-row
                (loop [j 0
                       row []]
                  (if (= j n)
                    row
                    (let [x-ij (sample unary)]
                      (when (> j 0)
                        (observe pairwise [x-ij (get row (dec j))]))
                      (when (> i 0)
                        (observe pairwise [x-ij (get (get rows (dec i)) j)]))
                      (recur (inc j) (conj row x-ij)))))]
            (recur (inc i) (conj rows next-row)))))))))
```