Solution

```
Initialize two data structures,
    one holding all positions p with the preferences of the hospital they correspond to,
    and one with all students s and their preferences

    while there remain positions that are not filled by a student

         choose such a position p

         let s be the highest ranked student in p's preference list
              to whom p has not offered a position

            if s is free
                (p, s) --> s fills position p
            else s has another position
                 if s prefers p to her current position
                    (p, s)
                 else s prefers her current position over p
                      OR p is a the same hospital as her current position
                     p remains open
                 endif
            endif
      endwhile
      return the set of matched pairs
```

Proof. Proof of running time:

In the case of this algorithm, each iteration consists of some position being offered to a student. Because there are p positions and s students, there can be at most ps iterations. Since we know that there are more positions p than s, we can call the upper bound running time p^2.

Proof that this returns a stable matching:

Suppose that s is assigned to p and s' is not assigned to a hospital but p prefers s' to s. This means that either p did not offer a position to s', or that p did offer a position but s' declined. According to the algorithm, p offers positions to students in order of p's preference, so if p is matched to s, but prefers s', it must be the case that p offered a position to s' before s. Since they are not paired, this means that s' must have declined the offer. But since s' is not paired with anyone, this also cannot be true, because students accept positions they are offered if they are unmatched. Therefore it cannot be the case that this algorithm produces matchings such that p prefers s' to its current match, but s' remains unmatched.

Suppose that s is assigned to p and s' is assigned to p' but p prefers s' and s' prefers p. This means that p offered a position to s, and s accepted. But if p preferred s' to s, then p must have proposed to s' before proposing to s. Therefore, s' must have rejected p in favor of its match p'. But we know that s' prefers p to p', so this also cannot be true. Therefore it cannot be that the algorithm produces this unstable matching. ☐