a)

(1) Suppose there comes a point when tenant *a* is free but has already proposed to every apartment's landlord. Each of the *n* apartments is matched at this point in time by definition of the stable marriage problem. Since the set of engaged pairs forms a matching, there must also be *n* engaged tenants at this point in time. But there are only *t* tenants total, and *a* is not engaged, so therefore there must be at least one apartment's landlord that the tenant has not proposed to.

(2) The set of engaged pairs always forms a matching. Let us suppose that the algorithm terminates with a free tenant *t*. At termination, it must be the case that *t* had already proposed to every apartment's landlord, for otherwise we would have immediately stopped the algorithm's progress. But this contradicts (1), which says that there cannot be a free tenant who has proposed to every apartment's landlord. Therefore there must always exist a perfect matching that is at least weakly stable.

b)

```
while(a tenant is free and that tenant has not proposed to all apartment options) {
        for(preferredApartment : tenantPrefs) {
                if(preferredApartment is available) {
                        matches[tenant] = preferredApartment;
                        break;
                }
                if(currentLandownerOfPreferredApartment prefers new tenant) {
                        matches[tenant] = preferredTenant;
                        matches[otherTenant] = null;
                        break;
                }
        }
}
return new Matching(currentMatch, matches);
```

c)

i. Terminates: Every iteration a tenant has proposed to at least one new apartment's landlord. Therefore each tenant will eventually propose to all landlords, invalidating the second condition of the while loop and exiting the algorithm.
ii. Correct result: There cannot be an unstable matching since any higher preference immediately replaces the current matching and continues to the next iteration of the loop. Since there cannot be an unstable matching produced and the algorithm terminates, then the result should be correct.

d)

$O(t \cdot n)$

This Big-Oh of $O(t \cdot n)$ is accurate because the main loop can only iterate over all of the tenants as many times as they have not proposed to their apartment's landlord. Therefore the worst-case time complexity is the number of tenants by the number of landlords.