Homework 5 - Stable Marriage

Ludvig Larsson

1 Implementation

In my implementation there always needs to be an odd number of processes provided upon execution `-np 2n+1`. To allow all men and women to get married they have to be equally many, as well as an additional process, monitor-like, that will keep count of women that have received at least one proposal. The program starts by letting processes share the same lines of code. They create profiles of opposing gender and rate them 1 to N. Afterwards the ratings are shuffled to introduce differences in execution trace (non determinism). Afterwards, the monitor, men and women will enter different functions until the program terminates since they perform disjoint operations. Men will propose to its highest ranked woman first, and then wait for a response (arbitrary amount of time). Once it has received the response, the returned value will either instruct the man to exit the proposal loop or propose to the second best rated woman. Women wait for proposals initially and temporarily accepts the proposing man. In case another proposal arrived to the same woman, she will compare the currently accepted man against the new proposal and possibly replace. Women's first received proposal will trigger a send from woman to the monitor process as we need to keep count of the total women in the program that has at least one proposal. The monitor saves each woman's rank to later notify them all when all women have been proposed to. The receive operation for women will differentiate between if it was a man proposing or if the monitor instructs the woman to quit waiting for proposals. Lastly, the woman who broke out of the receiving loop will notify the waiting man and program can successfully quit.

2 Performance

I have made sure that there are limited amount of messages being sent between processes, no polling is done to check for instruction to terminate for example and I have used the fully connected graph model to a certain extent. Execution trace is printed out as the program is running but it was very hard to synchronize as there are no locks involved, I did not have time to send printouts with timestamps to a file and then sort it before writing to standard output. It could also have been possible to let the processes use one terminal each for printing execution trace. The overall result is great I'd say and I'm satisfied!