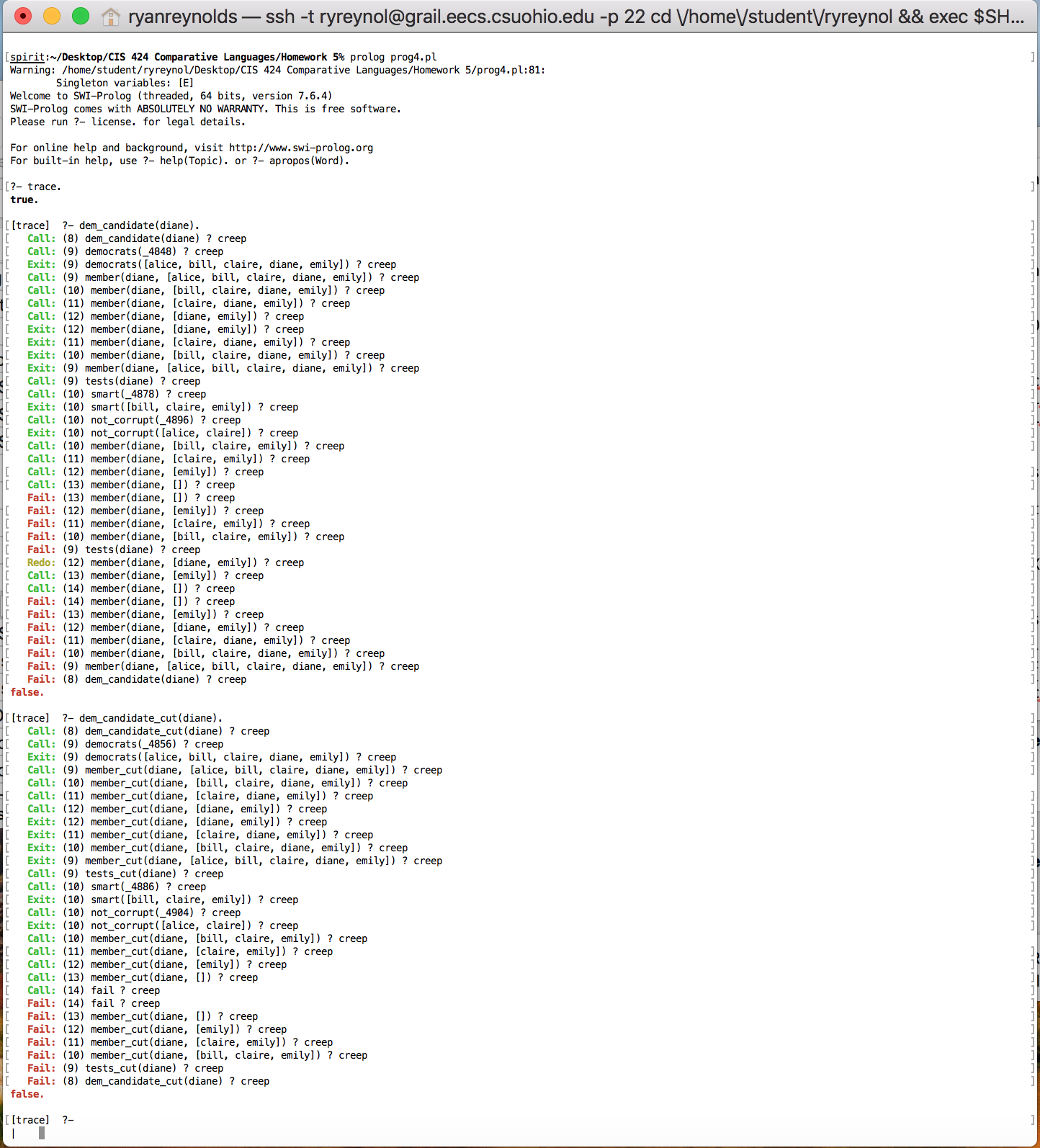
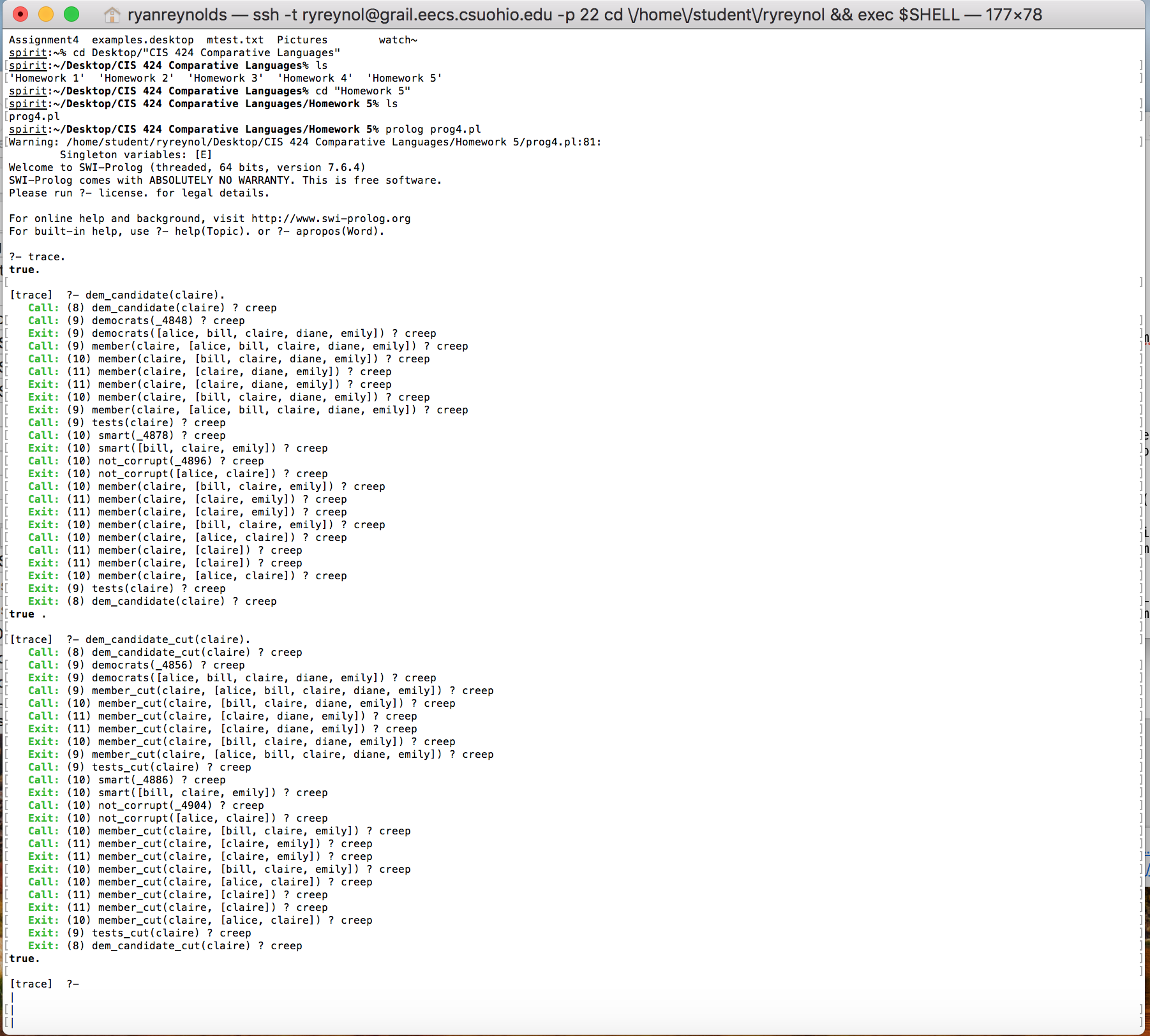
Question 4:

The screenshot below shows how a cut statement can improve performance by limiting backtracking. Diane is a democrat but she is not smart and is corrupt so she fails the good democrat test. Without the cut statement prolog determines she is not smart and returns to the dem\_candidate rule. Prolog backtracks to see if there is another ‘diane’ case in the democrats list, which is redundant for this application. Implementing a cut removes this backtracking. Essentially prolog is told to find a single match for each member\_cut call.

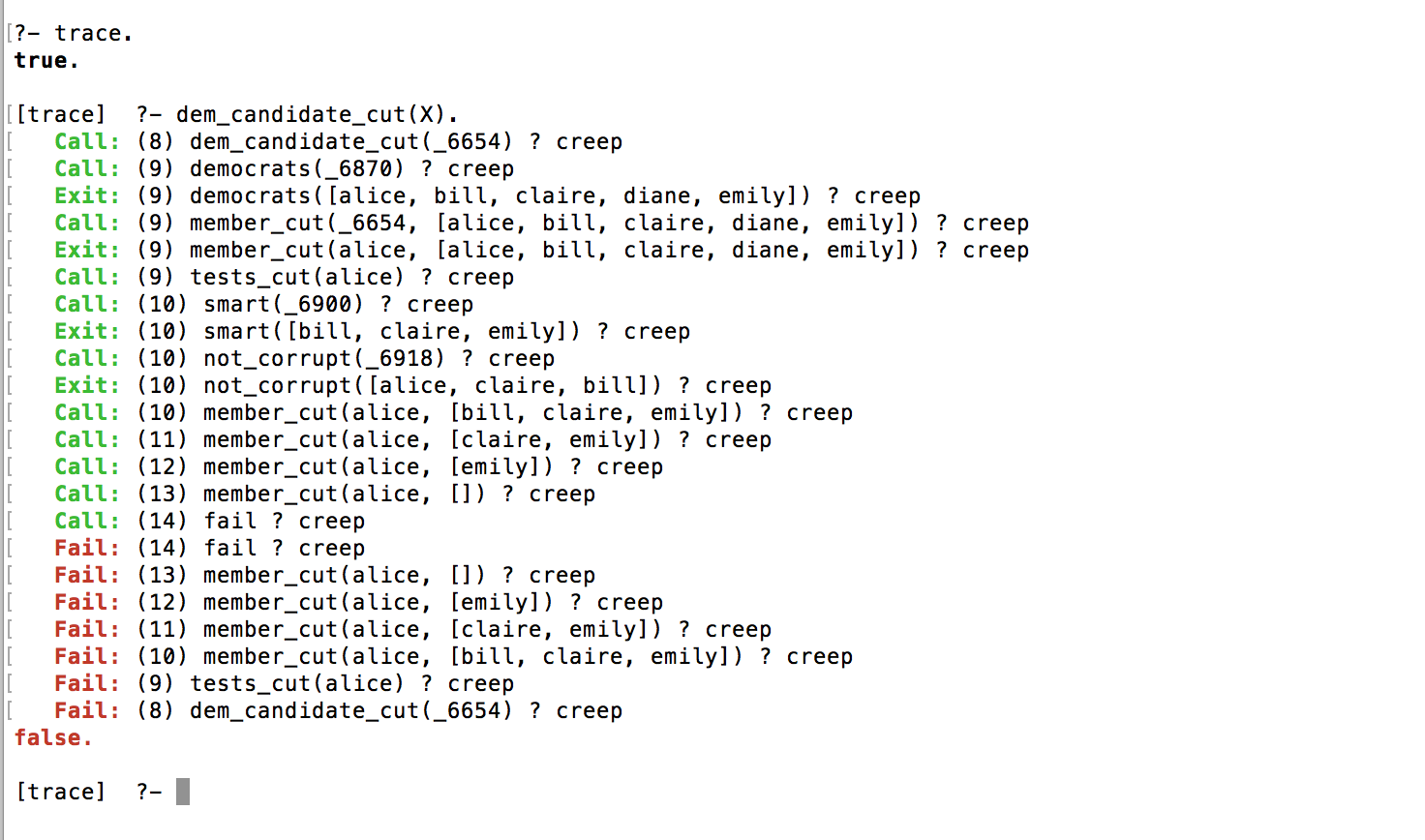


The below screenshot shows that for this case a cut in a successful query will not improve performance. This is because a successful query for this question does not require any backtracking.



Q4 General Cases:

With the cut statement the general test is limited to the first instance because no backtracking is allowed when checking if an element is a member of a list. Alice is checked and because Alice is not smart the query returns false and exits with a false. This shows how cuts statements can be dangerous and limit functionality.



Without a cut statement we are able to backtrack through the entire list and return all of the possible good democratic candidates. This is more dynamic but as shown below the process is costly. Moreover, it is unnecessary for this application because user’s will be querying by passing a single name to check.

