A4Q2

a)

- line 5,6: Data Hazard as $6 must load data from 0($1) before branch prediction on line 6 can start.
- Line 7, 8, 9: Data Hazard as multiplication instruction on line 9 depends on values being stored in $7 and $8 in lines 7 and 8 (respectively) first.
- Line 6, 11, 16 all have Control Hazards as fetching of the next branch depends on conditions if $6 = $0, $7 = $0, $6 != $0 before going to branches “endloop”, “endif” and “loop” respectively.
- Line 13, Data Hazard as value must be stored in 0($3) before it can be undergo sw into register $6
- Line 15, Data Hazard as need to wait on line 13 which stores value from 0($3) into $6, before value in $6 is ready to be stored into 0($1)

b)

Forwarding allows the result of an execution to be used right after it is computed, without having first stored it first. This forwarding can save stalls – an example of this would be in lines 7,8,9 where computation of slt stores the computed result in $7 and $8 in line 7 and 8 respectively, for multiplication in line 9. If this computation undergoes forwarding, we would save one stall for each slt computation as mult in line 9 would be able to use the values computed from line 7 and 8 immediately. Since there are 20 numbers, 40 stalls would be saved.

Static branch prediction predicts that if we are incrementing our program counter positively, then we predict that we should ignore it, and if we are going backwards i.e. Decrementing our program counter, then we take it. 20 stalls will be saved by line 6 as we predict that we do not take the increment in program counter to jump to endloop. No stalls would be saved by line 11 since we end up branching to end if 20 times anyway. 20 stalls would be saved by line 16 as it is correctly predicted 20 times that we should take the decrement in program counter to line 6. Thus 40 stalls would be saved in total using static branch prediction.