P1 A. The current state {C, F} has an error of 8. The states for the next step are {C}, {F}, {C, D, F}, {C, E, F}, {C, F, G}, {C, F, H} with errors 1, 10, 22, 16, 11, 9. The best neighbor is S = {C}, where the cost is Error(S) = max(18−18, 0) + max(20−19, 0) = 1.
As we're using hill-climbing, there is no better error smaller than Error({C}), thus in the next iteration no update will be made and the set {C} is the result.

B. The state space size is 2^N as each object can either be taken or not.
Let the current state have m objects then there are m objects to be deleted and (N-m) objects to be added. The maximum number of neighbors is N where at most (N-m)+m=N objects can be either added or deleted.

P2 1. A -> ¬(B ^ C) = ¬A V ¬(B ^ C) = ¬A V ¬B V ¬C
{¬A V ¬B V ¬C}

2. C <-> ¬(D V E) = (C -> ¬(D V E)) ^ (¬(D V E) -> C) = (¬C V ¬(D V E)) ^ (C V D V E)
= (¬C V (¬D ^ ¬E)) ^ (C V D V E) = (¬C V ¬D) ^ (¬C V ¬E) ^ (C V D V E)
{¬C V ¬D, ¬C V ¬E, C V D V E}

3. D -> B = ¬D V B
{¬D V B}

4. (D^E) -> ¬B ^ A = ¬(D ^ E) V (¬B ^ A) = ¬D V ¬E V (¬B ^ A)
= (¬D V ¬E V ¬B) ^ (¬D V ¬E V A)
{¬D V ¬E V ¬B, ¬D V ¬E V A}

5. D <-> E = (¬D V E) ^ (D V ¬E)
{¬D V E, D V ¬E}

P3 S0 = {C1: ¬A V ¬B V ¬C, C2: ¬C V ¬D, C3: ¬C V ¬E, C4: C V D V E, C5: ¬D V B,
C6: ¬D V ¬E V ¬B, C7: ¬D V ¬E V A, C8: ¬D V E, C9: D V ¬E}, try A = T;
S1 = {C1: ¬B V ¬C, C2: ¬C V ¬D, C3: ¬C V ¬E, C4: C V D V E, C5: ¬D V B,
C6: ¬D V ¬E V ¬B, C8: ¬D V E, C9: D V ¬E}, try B = T;
S2 = {C1: ¬C, C2: ¬C V ¬D, C3: ¬C V ¬E, C4: C V D V E,
C6: ¬D V ¬E, C8: ¬D V E, C9: D V ¬E}, C = F;
S3 = {C4: D V E, C6: ¬D V ¬E, C8: ¬D V E, C9: D V ¬E}, try D = T;
S4 = {C6: ¬E, C8: E}, contradicted and get back to S3;
S3 = {C4: D V E, C6: ¬D V ¬E, C8: ¬D V E, C9: D V ¬E}, try D = F;
S4 = {C4: E, C9: ¬E}, contradicted and get back to S3 then to S2 then to S1;
S1 = {C1: ¬B V ¬C, C2: ¬C V ¬D, C3: ¬C V ¬E, C4: C V D V E, C5: ¬D V B,
C6: ¬D V ¬E V ¬B, C8: ¬D V E, C9: D V ¬E}, try B = F;
S2 = {C2: ¬C V ¬D, C3: ¬C V ¬E, C4: C V D V E, C5: ¬D, C8: ¬D V E, C9: D V ¬E}, D = F;
S3 = {C3: ¬C V ¬E, C4: C V E, C9: ¬E}, try C = T;
S3 = {C3: ¬E, C9: ¬E}, E = F;
Return {A-T, B-F, C-T, D-F, E-F}.