Sample Written Answers PS4:

Distribution of points:
1) 5
4) 3
7) 3
9) 3
12) 6
Total: 20

1. a) O: yes. c=4 n0=1
   Omega: yes. c=1 n0=1
   Theta: yes. (it's in both others)
   b) O: yes. c=2 n0=1
   Omega: yes. c=1 n0=1
   Theta: yes. (it's in both others)
   c) O: yes. c=1 n0=1
   Omega: no. As long as c is positive - no matter what n0 is 3^n will always be greater than 2^n
   Theta: no. (it's not in both others)
   d) O: yes. c=2 n0=1
   Omega: no. Same reasoning as above - f increases much faster than g in all cases
   Theta: no (it's not in both others)
   e) O: no - debt (g) increases much faster than population
   Omega: yes - opposite of above
   Theta: no. (it's not in both others)

4. The baudot-to-string function performs char-to-string on each baudot-character and then performs list->string which cycles through each letter and reforms the string. The exact running time for this is n+n or 2n which is in both O(n) and Omega(n) and therefore our Theta running time is Theta(n).

7. In this function we're rotating each of w number wheels a total of r rotations. Since we know a single wheel rotation takes 1 unit time (in that it's a simple cons) then this function would have a running time of Theta(wr) - that is: r rotations over each of w wheels. In the traditional sense we could say w is our n (the size of our input) and r is a constant so in this case it could simplify to Theta(n). Either answer is acceptable for this homework. Notice the number of positions on the wheel has nothing to do with the calculations. If you think about it - more or less positions doesn't make rotating wheels any faster or slower and therefore it doesn't affect the overall running time.

9.The general idea of this function is that we're going to xor every bit of our n-digit(in our case 5-digit) baudot-character with the corresponding wheel in the wheel-list. Since xor takes 1 unit time and we perform it over n bits then the running time is both O(n) and Omega(n) and therefore also Theta(n).

12. Overall the running time is p^w where n is the number of positions and w is the effective number of wheels (recall brute had 125 possibilities and we had 3 effective wheels and 5 positions - 5^3 = 125)

a. In this case the positions is the base - so 61^3

b. In this case the number wheels is the exponent so it would be 61^(2w+1) where w is the number of s and k wheels and the extra 1 is for the m wheel

c. Many possibilities -

1) Change the respective number of S and K wheels
2) Encrypt it multiple times (probably the best solution)
3) etc...most anything reasonable was OK here