1. No. From the definition f(n)=O(1) means for real number C and N, there exist f(n)<=C*1 (n>N) so we get f(n)<=C. and f(n)=O(n/100) means there exist f(n)<=C*n/100 (n>N), we know that when n>N, there exist real number which satisfied C<C*n/100, so f(n)=O(1)<O(n/100).

2. No. From the definition g(n)=O(n) means for real number C and N, there exist g(n)<=C*n (n>N) so we get g(n)<=Cn, and g(n)=O(100) means there exist g(n)<=C*100 (n>N), we know that when n>N, Cn will bigger than C*100 so we cannot claim g(n) = O(100).

3. Yes. A(n)=f(n)+g(n)=O(1)+O(n) ,so we need to proof that O(1)+O(n)=O(n) .
   O(1)+O(n)=O(n) =>
   O(1)+O(n)<Cn;
   A+Bn<Cn;
   Because A<A*n in this domain.
   So we just get A+B*n<A*n+B*n<C*n => (K)*n<C*n
   So A(n)=O(n).

4. Yes. B(n)=f(n)g(n)=O(1)*O(n), so we need to proof that O(1)*O(n)=O(n)
   O(1)*O(n)=O(n) =>
   O(1)*O(n)=C*n;
   A*B*n=C*n; => K*n=C*n
   So B(n)=O(n);

Bonus

1. C(n)=O(1) means f(O(n))=O(1). We know g(n)=O(n) means g(n) can be written by A*n+B, that means the complexity of g(n) is A*n+B. if we set A*n+B into code f(n) , because the complexity of f(n)=O(1), so the complexity of f(O(n))=M(A*n+B)=O(n)=\=O(1).
2. D(n)=O(1) means g(O(1))=O(1). We know f(n)=O(1) means the complexity of f(n) is a constant C, then we input that constant to the function g(n), because C is constant, the complexity of g(C) must be constant so D(n)=O(1).