You Do
Some Big-Oh Analysis
Give a Big_Oh analysis of the running time of each function.
1. // sums the numbers from 1 to n
   int A(int n) {
       int sum = 0;
       for (int i=1; i <= n; i++)
           sum += i;
       return sum;
   }

A. O(log n)
B. O( n )
C. O( n^2 )
D. O(n+1)
int A(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        sum += i;
    return sum;
}

Answer B: This performs n additions. O(n).

Answer D: O(n+1) is also correct, but we usually simplify the order of growth as much as possible.
2.

```c
int B(int n) {
    int sum = 0;
    for (int i=1; i <= 2*n; i++)
        for (int j=0; j < 5; j++)
            sum += j;
    return sum;
}
```

Is this

A. $O(\log n)$
B. $O(n)$
C. $O(n^2)$
D. $O(5*2*n)$
2.

```c
int B(int n) {
    int sum = 0;
    for (int i=1; i <= 2*n; i++)
        for (int j=0; j < 5; j++)
            sum += j;
    return sum;
}
```

Analysis: The inner for-loop (on j) always adds 5 numbers together, and the outer loop (on i) does this 2*n times. So this is $O(5\times2\times n) = O(n)$. Answers B and D are both correct, but answer A is better.
3.

```c
int C(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        for (int j=0; j <= n; j++)
            sum += j;
    return sum;
}
```

Is this

A. $O(n)$
B. $O(n^2)$
C. $O(n^n)$
D. The answer depends on what $n$ is.
3.

```c
int C(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        for (int j=0; j <= n; j++)
            sum += j;
    return sum;
}
```

Analysis: The inner loop (on j) runs n steps as for each value of i from 1 to n. Altogether this does n+n+n+ ...+ n steps. Those numbers sum to n*n, so this is O(n^2).
4. int D(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        sum += i*i;
    for (int j=0; j < n; j++)
        sum -= j;
    for (int k = 0; k < 2*n; k++)
        sum = sum * k;
    return sum;
}

A. O(n)  
B. O(n^2)  
C. O(n^3)  
D. O(n^n)  

4.

```c
int D(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        sum += i*i;
    for (int j=0; j < n; j++)
        sum -= j;
    for (int k = 0; k < 2*n; k++)
        sum = sum*k;
    return sum;
}
```

Analysis: Note that the loops are sequential, not nested. The loop on i does n additions. After that is finished the loop on j does n subtractions. Then the loop on k does 2*n multiplications. Altogether there are 4*n steps.

This is $O(n)$