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

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

Analysis: This performs $n$ additions. $O(n)$
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;
}
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 $O(5 \times 2n) = O(n)$
3.

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

Analysis: This time the inner loop (on j) runs \(i\) steps as \(i\) goes from 1 to \(n\). Altogether this does 1+2+3+ ...+ \(n\) steps. Those numbers sum to \(n^2(n+1)/2\), so this is \(O(n^2)\).
int D(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        sum += i;
    for (int j=0; j < n; j++)
        sum -= j;
    return sum;
}
4.

```c
int D(int n) {
    int sum = 0;
    for (int i=1; i <= n; i++)
        sum += i;
    for (int j=0; j < n; j++)
        sum -= j;
    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. Altogether, 2*n operations are done, so this is $O(2*n) = O(n)$. 