Q1	``` double sum_triples(double array[], int n) { //n: size of the array. Assume n is divisible by 3 double sum=0; for (int i=0; i<n; i=i+3) sum = sum + array[i]; return sum; } ```	RUNS: 1 RUNS: (n / 3) + 1 RUNS: n / 3 RUNS: 1
Q2	``` double sum_exponentials(int n) { //n is a power of 2, i.e., n=2k or k=log2n int sum=0; for (int i=1; i<n; i=i*2) sum = sum + i; return sum; } ```	RUNS: 1 RUNS: log2n + 1 RUNS: log2n RUNS: 1
Q3	``` for (int i=0; i<10; i++) for (int j=0; j<n; j++) cout << i << "," << j <<endl; ```	RUNS: 11 RUNS: 10n + 10 RUNS: 10n
Q4	``` for (int i=0; i<n; i++) for (int j=0; j<n; j++) cout << i << "," << j <<endl; ```	RUNS: n + 1 RUNS: n^2 + n RUNS: n^2
Q5	``` for (int i=0; i<n; i++) //assume n is divisible by 2 for (j=n/2; j>i; j--) sum = i+j; ```	RUNS: n + 1 RUNS: $(n^2 + 10n)$ / 8 RUNS: $(n^2 + 2n)$ / 8
Q6	``` double sum_matrix(double matrix[][], int m, int n) { //m: num of rows; n: num of cols double sum=0; for (int i=m-1; i>=0; i--) { for (int j=n-1; j>=0; j--) { sum = sum + matrix[i][j]; } } return sum; } ```	RUNS: 1 RUNS: m + 1 RUNS: nm + m RUNS: nm RUNS: 1