Name: \_\_\_\_\_Grace Miguel\_\_\_\_\_\_\_\_\_ Date: \_\_\_2/19/21\_\_\_\_\_\_\_

Pledge: \_\_I pledge my honor that I’ve abided by the Stevens Honor Code\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Give the complexity of the following functions. Choose the most appropriate notation from among , and .

1. **void** **function1**(**int** n) {

**for** (**int** i = 1; i <= n; i++) { // runs n times

**for** (**int** j = i; j <= n; j += 2) { // runs n times

cout << "\*"; // constant time

}

}

}

C\*n\*n =(n^2)

Answer: \_\_\_\_\_\_\_\_

1. **void** **function2**(**int** n) {

**int** count = 0; // constant c0

**for** (**int** i = 1; i \* i \* i <= n; i++) { // times

count++; // constant

}

cout << count; // constant c1

}

C0 + c1(n/3) = (//)

Answer: \_\_()\_\_\_\_\_\_

1. **void** **function3**(**int** n) {

**int** count = 0; // constant c0

**for** (**int** i = 1; i \* i <= n; i++) { //sqrt(n)

**for** (**int** j = 1; j + n/2 <= n; j++) { //n/2

**for** (**int** k = 1; k <= n; k \*= 2) { //logn

count++; //constant c1

}

}

}

cout << count; //constant c2

}

C0 + c1\*sqrt(n)\*n/2\*logn = (n)

Answer: \_\_\_\_(n^(3/2)logn)\_\_\_\_

1. **void** **function4**(**int** n) {

**int** count = 0; // constant c0

**for** (**int** i = n/2; i <= n; i++) { //n

**for** (**int** j = 1; j <= n; j \*= 2) { //logn

**for** (**int** k = 1; k <= n; k \*= 2) { //logn

count++;

}

}

}

cout << count;

}

Answer: \_\_ ()\_

1. **void** **function5**(**int** n) {

**if** (n % 2 == 0) { //constant

**return**;

}

**for** (**int** i = 1; i <= n; i++) { n

**for** (**int** j = 1; j <= n; j++) { n

cout << "\*"; constant

**break**;

}

}

}

Answer: \_\_\_\_O(\_watch out for the break statement, the inner loop is really \_ (1). For even numbers the algorithm runs in constant time, so we cannot have .

1. **void** **function6**(**int** n) {

**int** count = 0; //c0

**for** (**int** i = 1; i <= n/2; i++) { //n/2

**for** (**int** j = 1; j <= n/3; j++) { //n/3

**for** (**int** k = 1; k <= n/4; k++) { //n/4

count++; //c1

}

}

}

cout << count; //c2

}

Answer: \_\_ )

1. **void** **function7**(**int** n) {

**for** (**int** i = 1; i <= n; i++) { //n

**for** (**int** j = 1; j <= n; j += i) { //n-1

cout << "\*";

}

}

}

Answer: \_\_)\_\_\_\_\_\_

1. **void** **function8**(**int** n) {

**int** i = 1, s = 1;

**while** (s <= n) {

i++; //c

s += i; //c

cout << "\*";

}

}

Answer: \_\_\_\_\_\_\_\_

This loop terminates when s=1+2+..+I, or becomes > n.

1. Processing Arrays
   1. Suppose you have an unsorted array of integers of length and want to sum all the elements inside it. What is the running time of your algorithm? \_\_\_\_\_\_\_\_
   2. Suppose you have an unsorted array of integers of length and want to determine if all the values inside are positive. What is the running time of your algorithm? \_\_\_\_\_\_\_\_
   3. Suppose you have a sorted array of integers of length and want to determine the median value. What is the running time of your algorithm? \_\_\_O(1)\_\_INCORRECT\_\_\_**CORRECT ANSWER:**

If(n %2 ==0) //c

Then do (arr[(n/2)-1]+arr[n/2])/2 //c

Else

Return arr[n/2]

1. \_\_\_\_ **T** / F

If true, prove it by giving *integral* values for the required constants , , and . Choose the tightest values possible for the and constants. If false, show the contradiction.

C1n^2 <= 5n^2 + n^2 + n^2 <= c2n^2

= 5

=6

= 6