

SOLUTION

CSI 33 Fall 2026 Test 1 (Ojakian)

Instructions:

You may use your one cheat sheet. Each question is worth 10 points (scaled to 100). Show all work for full credit, and for any hope of partial credit! For questions that ask you to write code, you do not need to show work unless the question asks you to (your code is your work!), but on other questions, you **must show your work to get full credit**. Your programs should work without importing any special packages (in Python you should not import anything, while in C++, just the standard ones- iostream, string, vector).

No electronic devices allowed. If you use one, or one is even out, at a minimum you will get a zero on the exam.

1. What does the following C++ code print? What "delete" command should be at the end of the code?

```
int a = 7;
int *x, *y;
x = &a; implies x → a: 7
y = new int; implies y → 2
*y = 2;
cout << "a: " << a << ", *x: " << *x << ", *y: " << *y << endl;

*x = 5; implies x → a: 5
*y = *y + 10; implies y → 12
cout << "a: " << a << ", *x: " << *x << ", *y: " << *y << endl;

a = a + 3; → x → a: 8
*y = *x + 1; → y → 9
x = y; → x → 9
a = a + 2; → a: 10
*x = *x + *x; → x → 18
cout << "a: " << a << ", *x: " << *x << ", *y: " << *y << endl;
```

→ delete y;

print #1

a: 7, x: 7, y: 2

print #2

a: 5, x: 5, y: 12

→ a: 10, x: 18, y: 18 print #3

2. What do the following two Python programs print?

(a) L = [2, 3, 4, 4, 2]

print(L[2])

if 3 < L[4]:

print(L[0])

else:

print(L[0] + 1)

(b) A = 0

for x in range(100):

A = A + 2

print(A)

→ 4

→ since 3 < 2 ← L[4] is False

→ Do: → 3

(since 2 + 1 = 3)

↑
L[0]

← Add 2 to 0

100 times

(i.e. Add 200)

→ Thus, 200

3. What do the following two C++ programs print?

(a)

```
int i;
i = 0;
vector<int> A = {2, 16, 0, 2};
cout << "First: " << A[i] << endl;
i = i + 2;
cout << "Second: " << A[i] << endl;
cout << "Third: " << A[i] + 1 << endl;
```

First: 2 since $A[0] = 2$

Second: 0 since i now = 2 and $A[2] = 0$

Third: 1 since $A[2] + 1 = 0 + 1 = 1$

(b)

```
int k = 1;
int n = 16;
while (k < n) {
    cout << k << endl;
    k = 2 * k;
}
```

1
2
4
8

since double k each iteration but when $k = 16$, $16 < 16$ False so nothing printed. But final k

```
cout << "Final k: " << k << endl;
cout << "Final n: " << n << endl;
```

Final k: 16

Final n: 16 since n never changed

4. Write a Python **function** that takes one positive integer n as input, and **returns** the sum of the odd numbers between 1 and n , *not including* n . For example: if the input is 5, then the function should output 4 (since the odd numbers less than 5 are 1, and 3, and $1 + 3 = 4$).

```
def F(n):
```

```
    A = 0
```

```
    k = 1
```

```
    while (k < n):
```

```
        A = A + k
```

```
        k = k + 2
```

```
    return A
```

5. Write a C++ **function** that takes one integer n as input, and **returns** the integer 1 if n is greater than zero, returns -1 if n is less than zero, and returns 0 if n is equal to zero.

```
int F(int n) {  
    int x = 0;  
    if n < 0 { x = -1; }  
    if n > 0 { x = 1; }  
    return x;  
}
```

6. Consider the Binary Search algorithm applied to a sorted list of length n .

- (a) What is the run time complexity (i.e. Theta analysis) of the algorithm? (just give the answer on this part).
(b) Suppose the list is [3, 5, 6, 7, 12, 16, 20].
i. Carry out the Binary Search algorithm on the list, when searching for the number 12, showing every step.
ii. Carry out the Binary Search algorithm on the list, when searching for the number 4 (not in the list), showing every step.

Note if needed: Use the same convention we did in class - when finding the middle element of an even length list, choose the right position from the 2 possible middle positions.

a) $\Theta(\log n)$

b) i) • Middle : 7

• Consider right half: [12, 16, 20] since $12 > 7$

• Middle 16

• Consider left half [12] since $12 < 16$

• List [12] only contains 12. FOUND.

ii) • Middle : 7

• Consider left half [3, 5, 6] since $4 < 7$

• Middle : 5

• Consider left half [3] since $3 < 4$

• Length 1 list [3], but $3 \neq 4 \Rightarrow$ NOT Found

7. Consider the following two Python programs, where for each one, the input size is n . Carry out a theta analysis for each program.

(a) `count = 0`

`for i in range(n):`

`for j in range(50):`

`count += 1`

$\leftarrow$ n steps

$\leftarrow$ For each, do

2 lines 50 times

so 100 steps

$\left. \begin{array}{l} \leftarrow n \text{ steps} \\ \leftarrow \text{For each, do} \\ \text{2 lines 50 times} \\ \text{so 100 steps} \end{array} \right\} \rightarrow 100n \rightarrow \text{steps}$

$\rightarrow \boxed{\Theta(n)}$

(b) `count = 0`

`for i in range(n):`

`for j in range(2n):`

`count += 1`

$\leftarrow$ n steps

$\leftarrow$ for each $2n$ times

do 2 lines

so $4n$ steps

$\left. \begin{array}{l} \leftarrow n \text{ steps} \\ \leftarrow \text{for each } 2n \text{ times} \\ \text{do 2 lines} \\ \text{so } 4n \text{ steps} \end{array} \right\} \rightarrow n \cdot 4n$

$= 4n^2 \rightarrow$

$\rightarrow \boxed{\Theta(n^2)}$

8. Recall that for functions $f(n)$ and $g(n)$, $f(n)$ is $O(g(n))$ if there exist constants C and n_0 such that $f(n) \leq C \cdot g(n)$ for all $n \geq n_0$. For each of the following cases, show that $f(n)$ is $O(g(n))$ by finding an appropriate C and n_0 .

(a) $f(n) = 20n$ and $g(n) = 4n$

$$20n \leq 5 \cdot 4n \text{ for } n \geq 1$$

$$\text{so } \boxed{C = 5} \text{ and } \boxed{n_0 = 1}$$

(b) $f(n) = 20n + 10$ and $g(n) = 4n$

$$20n + 10 \leq 8 \cdot 4n \text{ for } n \geq 1 \rightarrow \text{so } \boxed{C = 8} \text{ and } \boxed{n_0 = 1}$$

(since at $n=1$: $20(1)+10 \leq 8 \cdot 4(1)$)
and " $32n$ " is steeper line

(c) $f(n) = 5n^2 + n$ and $g(n) = n^3$

$$5n^2 + n \leq 5n^2 + n^2 \leq 6n^2 \leq 6n^3 \text{ for } n \geq 1$$

$$\text{so } \boxed{C = 6} \text{ and } \boxed{n_0 = 1}$$

9. For each pair of functions, state which has a faster growth rate, and find an integer n where the faster function is in fact larger. Do the calculation to show that your n value works (your choice of n and your showing that your n works, may involve unevaluated numerical expressions, for example 3^2 is fine instead of 9).

(a) n^3 versus $0.5n^4$

$$\begin{aligned} n=3 &\rightarrow 3^3 = 27 \\ &\rightarrow (0.5) 3^4 = (0.5)(81) = 40.5 \end{aligned}$$

$0.5n^4$ faster

(b) 2^n versus $10n$

$$\begin{aligned} n=6 &\rightarrow 10 \cdot 6 = 60 \\ &\rightarrow 2^6 = 64 \end{aligned}$$

2^n faster

(c) $n \log_2 n$ versus $9n$

$$\begin{aligned} n=2^{10} &\rightarrow 9 \cdot 2^{10} \\ &\rightarrow 2^{10} \cdot \log_2 2^{10} = 10 \cdot 2^{10} \end{aligned}$$

$n \log n$ faster

10. Write the definition of a Python class *Positive* containing:

- (a) A constructor that takes one input and initializes an instance variable **val** of type int, to the given input UNLESS the input is negative, in which case **val** is initialized to 0.
- (b) A method called *Get* that has no parameters and that returns the value of the instance variable **val**.
- (c) A method called *Dec* that has no parameters, and decreases **val** by 1, unless **val** would become negative; in that case it stays at zero.

class Positive:

def __init__(self, x):

self.val = x

self._zeroOut()

def _zeroOut(self):

if self.val < 0

self.val = 0

def Get(self):

return self.val

def Dec(self):

self.val -= 1

self._zeroOut()