# Page 1

Wednesday, April 26, 2023 8:23 PM

**1** With the pseudo-code below, determine the final value of sum when n=23.

**Algorithm 1**

```
procedure
    sum = 0
    for (i=2, i<n, i=i+3) do
        if i mod 2 == 0
          sum = sum + i
```

2 → 0+2=2
5
8 → 2+8=10
11
14 → 10+14=24
17
20 → 24+20=44
23

44

**2** Use the pseudo-code below to answer the following question.

(a) Without working through the pseudo-code, how many times with val be updated in Line 7?

(b) What is the final value of val.

**Algorithm 2**

```
1: procedure
2:    dFour = [1, 2, 3, 4]
3:    dSix = [1, 2, 3, 4, 5, 6]
4:    val = 0
5:    for (x in dFour) do
6:       for (y in dSix) do
7:          val = val + x·y
```

$0 + 1\cdot1 + 1\cdot2 + 1\cdot3 + 1\cdot4 + 1\cdot5 + 1\cdot6$
$+ 2\cdot1 + 2\cdot2 + 2\cdot3 + 2\cdot4 + 2\cdot5 + 2\cdot6$
$+ 3\cdot1 + 3\cdot2 + 3\cdot3 + 3\cdot4 + 3\cdot5 + 3\cdot6$
$+ 4\cdot1 + 4\cdot2 + 4\cdot3 + 4\cdot4 + 4\cdot5 + 4\cdot6 = 210$

a) 4 loops · 6 loops = 24
b) 210

# Page 2

Wednesday, April 26, 2023 8:38 PM

**3** Using the pseudo-code below, what is the ending value for count when `n = 100`? When `n = 10,000`? When `n =` $10^{42}$?

**Algorithm 3**

```
procedure
    count=0
    while (n>1) do
        count++
        n = n/10
```

$n_1 = 100 = 10^2 \rightarrow count_1 = \underline{2}$

$n_2 = 10{,}000 = 10^4 \rightarrow count_2 = \underline{4}$

$n_3 = 10^{42} \rightarrow count_3 = \underline{42}$

**4** This question assumes familiarity with modular arithmetic discussed in Unit 2. In the pseudo-code below, Line 7 performs string addition (concatenation), for example: `"a" + "b" = "ab"`. Even if the addends look like numbers, it will still do string addition: `"2" + "3" = "23"`. Answer the questions below.

a. What is the output of the algorithm when $x = 99$ and $b = 2$?

b. What is the output of the algorithm when $x = 1642$ and $b = 8$?

**Algorithm 4**

```
1: procedure EXPAND(x, b)
2:     Input:  integers x and b
3:     Output:  outVal, x written in base b
4:
5:     outVal = an empty string
6:     while (x>0) do
7:         outVal = string(x mod b) + outVal
8:         x = x div b
9:     return(outVal)
```

a) x = 99 b = 2

- (99 div 2 = 49 , 99 mod 2 = 1) → (49 div 2 = 24 , 49 mod 2 = 1) → (24 div 2 = 12 24 mod 2 = 0) → (12 div 2 = 6 , 12 mod 2 = 0) → (6 div 2 = 3 6 mod 2 = 0) → (3 div 2 = 1 3 mod 2 = 1) → (1 div 2 = 0 1 mod 2 = 1)

Blank String

- OutVal = (1+ "") → (1+1) → (0+11) → (0+011) → (0+0011) → (1+00011) → (1+100011)

  = <u>1100011</u>

b) x = 1642 b = 8

- (1642 div 8 = 205 1642 mod 8 = 2) → (205 div 8 = 25 205 mod 8 = 5) → (25 div 8 = 3 25 mod 8 = 1) → (3 div 8 = 0 3 mod 8 = 3)

- OutVal = <u>3152</u>

# Page 3

Wednesday, April 26, 2023 9:09 PM

**5** Order each function according to their growth from slowest to fastest (This is the same as execution speed from fastest to slowest for an algorithm).

(a) $n \cdot \log n$

(b) $\log_{10} n$

(c) $10^n$

(d) $n^{10}$

(e) $10n$

(f) $100$

(g) $n^2$

f) 100 b) $\log_{10} n$ e) $10n$ a) $n\log n$ g) $n^2$ d) $n^{10}$ c) $10^n$

**6** The provided expressions are the processing time of an algorithm for problems of size $n$. For each expression, find the *lowest possible* Big-$\mathcal{O}$ complexity stated in simplest form.

(a) $100n^2 + 0.0002n^3 + 10,000$

(b) $20n + 2n\log_{10} n + 30$

(c) $\log_{10} 5 + \log_2 7$

(d) $5n + \sqrt{n}$

(e) $0.01(2^n) + n^5 + \ln n$

(f) $100 \cdot 4^n + 200 \cdot 3^n$

a) $O(n^3)$ b) $O(n\log n)$ c) $O(1)$ d) $O(n)$ e) $O(2^n)$ f) $O(4^n)$

**7** $f(x) = 6x\log_4 x$ is which of the following? **Mark all that apply.**

○ $\mathcal{O}(6)$ ○ $\mathcal{O}(\log x)$ ○ $\mathcal{O}(6\log x)$ ○ $\mathcal{O}(x\log x)$ ○ $\mathcal{O}(x^2)$ ○ $\mathcal{O}(2^x)$

Since Big-O is upperbound, $f(x) \in O(x\log x) \subset O(x^2) \subset O(2^x)$, so

$O(x\log x)$, $O(x^2)$, $O(2^x)$

**8** Let $h(x) = 6x^4 + 2x^3 + x + 100$. Which of the following is true? **Mark all that apply.**

○ $h(x) \in \mathcal{O}(x^4)$ ○ $h(x) \in \Theta(x^4)$ ○ $h(x) \in \Omega(x^4)$

$O(x^4)$ upper bound $\Omega(x^4)$ lower bound $\Theta(x^4)$ tight bound