# CISC-235*
# Test #1
# January 24, 2020

Student Number (Required) _____

Name (Optional) _____

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, but will not be re-marked under any circumstances.

The test will be marked out of 50.

| | |
|---|---|
| Question 1 | /10 |
| Question 2 | /16 |
| Question 3 | /14 |
| Question 4 | /10 |
| | |
| | |
| **TOTAL** | /50 |

☐ By writing my initials in this box, I authorize the disposal of this test paper if I have not picked it up by April 15, 2020.

*"Fill your house with stacks of books, in all the crannies and in all the nooks"*

*--- Dr. Seuss*

**Question 1 (10 marks)**

(a) [5 marks] Prove the following statement:

If $f(n) \in \Theta(g(n))$, then $g(n) \in \Theta(f(n))$

Solution:

$$
\begin{aligned}
f(n) \in \Theta(g(n)) &\Rightarrow f(n) \in O(g(n)) \text{ and } f(n) \in \Omega(g(n)) \\
&\Rightarrow f(n) \leq c_1 * g(n) \text{ and } f(n) \geq c_2 * g(n) \\
&\qquad \text{for some } c_1, c_2 > 0 \quad \forall \text{ large n} \\
&\Rightarrow g(n) \geq \frac{1}{c_1} * f(n) \text{ and } g(n) \leq \frac{1}{c_2} * f(n) \\
&\Rightarrow g(n) \in \Omega(f(n)) \text{ and } g(n) \in O(f(n)) \\
&\Rightarrow g(n) \in \Theta(f(n))
\end{aligned}
$$

Marking:

| | |
|---|---|
| Sound proof | 5 marks |
| Proof with a small error | 4 marks |
| Proof with major flaw, but shows understanding of $O, \Omega, \Theta$ | 3 marks |
| Random words and symbols | 1 mark |

(b) [5 marks] Prove the following statement:

If $f(n) \in \Theta(g(n))$ and $g(n) \in \Theta(h(n))$, then $f(n) \in \Theta(h(n))$

Solution:

$$
\begin{aligned}
& f(n) \in \Theta(g(n)) \text{ and } g(n) \in \Theta(h(n)) \\
& \quad \Rightarrow f(n) \in O(g(n)) \text{ and } g(n) \in O(h(n)) \\
& \quad \Rightarrow f(n) \leq c_1 * g(n) \text{ and } g(n) \leq c_2 * h(n) \\
& \qquad\qquad\qquad\qquad \text{for some } c_1, c_2 > 0 \quad \forall \text{ large n} \\
& \quad \Rightarrow f(n) \leq c_1 * c_2 * h(n) \\
& \quad \Rightarrow f(n) \in O(h(n))
\end{aligned}
$$

AND

$$
\begin{aligned}
& f(n) \in \Theta(g(n)) \text{ and } g(n) \in \Theta(h(n)) \\
& \qquad \Rightarrow f(n) \in \Omega(g(n)) \text{ and } g(n) \in \Omega(h(n)) \\
& \qquad \Rightarrow f(n) \geq c_1 * g(n) \text{ and } g(n) \geq c_2 * h(n) \\
& \qquad\qquad\qquad\qquad \text{for some } c_1, c_2 > 0 \quad \forall \text{ large n} \\
& \qquad \Rightarrow f(n) \geq c_1 * c_2 * h(n) \\
& \qquad \Rightarrow f(n) \in \Omega(h(n))
\end{aligned}
$$

THEREFORE

$$f(n) \in \Theta(h(n))$$

Marking:

Same as for Part (a)

**Question 2 (16 marks)**

Determine the $O$, $\Omega$ and (if possible) $\Theta$ classification of the following code segments (which are written in generic pseudo-code):

(a) [8 marks]

```
n = read_int()
A = array[1..n]
B = array[1..n]
for i = 1 to n:
      A[i] = read_int()
for i = 1 to n:
      B[i] = read_int()
for i = 1 to n:
      for j = 1 to n:
            print A[i]+B[j]
```

Solution:

$O(n^2)$

$\Omega(n^2)$

$\Theta(n^2)$

Marking:

| | |
|---|---|
| $O$ classification | 3 marks |
| $\Omega$ classification | 3 marks |
| $\Theta$ classification | 2 marks if it fits with their O and $\Omega$ |

(b) [8 marks]

```
n = read_int()
if (n mod 3 == 0):

        for i = 1 to n*n:
                print i

for i = 1 to n:
        print i
```

Solution:

$O(n^2)$

$\Omega(n)$

$\Theta$ : no classification

Marking: Same as for Part (a)

**Question 3 (14 marks)**

Let `Stack` be a class that implements the stack data structure. Each instance of `Stack` has three defined methods:

| | |
|---|---|
| `push(x)` | add x to the top of the stack |
| `pop()` | remove and return the top value of the stack |
| `isEmpty()` | return true iff the stack is empty |

and a single accessible attribute:

| | |
|---|---|
| `count` | the number of items currently in the stack |

New stacks can be created like this:

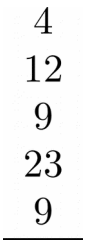
```
S = new Stack()
```

(a) [9 marks]

Let S be a stack containing integer values.

Write an algorithm that will print the number of values in S that are greater than all the values above them in the stack. After your algorithm finishes, S should be back in the same condition as when you started.

Example: If S is




then the algorithm should print "3" because the 4 , the 12 and the 23 are each greater than every value above them in the stack.

Your algorithm may declare and use simple variables and additional stacks, but it must not use arrays, linked lists, hash tables or other structures.

Page for answering Question 3 Part (a)

Solution:

```
if S.isEmpty():
        counter = 0
else:
        temp = new Stack()
        counter = 1
        max_so_far = S.pop()
        temp.push(max_so_far)
        while not S.isEmpty():
                next = S.pop()
                temp.push(next)
                if next > max_so_far:
                        counter += 1
                        max_so_far = next
        while not temp.isEmpty():
                S.push(temp.pop())        # this can be done in 2 steps

print counter
```

Marking:

| | |
|---|---|
| Correct algorithm | 9 marks |
| Correct with minor error (eg using "pop" instead of "push") | 7 marks |
| Correct with major logic error (eg potential crash due to empty stack, items in stack not in original order) | 5 marks |
| Severely incorrect (eg student has no idea what to do, but seems to know what a stack is) | 3 marks |
| Random words and symbols | 1 mark |

(b) [5 marks]

Determine the $O(\ )$, $\Omega(\ )$, and $\Theta(\ )$ classification of your algorithm, if possible.

Solution:

Let n be the number of values on the stack. Each loop iterates through all n values, doing a constant amount of work on each iteration. There are no special cases where the algorithm does more iterations or fewer iterations. Thus the algorithm is in $O(n)$ and also in $\Omega(n)$. Therefore it is in $\Theta(n)$

Marking:

I forgot to give the "explain your answer" instruction so students can get full marks just by stating the answer.

| | |
|---|---|
| Correct answer for their algorithm (which may not be the same as mine) | 5 marks |
| Incorrect answer that shows some understanding of the question | 3 marks |
| Incorrect answer without evidence of understanding | 1 mark |

**Question 4 (10 marks)**

Let A and B be one-dimensional arrays of integers, each containing n values.

You may assume that there are no duplicates in A and no duplicates in B – however some integers may be present in both arrays.

Write an algorithm that will print only the values that appear in both arrays.

For example if A =

| | | | | | | |
|---|---|---|---|---|---|---|
| 18 | 7 | 93 | 5 | 59 | 12 | 41 |

and B =

| | | | | | | |
|---|---|---|---|---|---|---|
| 93 | 85 | 12 | 18 | 34 | 25 | 66 |

Then the output should be $18 \quad 93 \quad 12$ (but not necessarily in that order)

Your algorithm should be as efficient as possible.

You may assume that there is a built-in `sort` function that has $O(n * \log n)$ complexity, and a built-in `binary_search` function that has $O(\log n)$ complexity. Whether or not you use them is up to you.

Write your answer on the next page.

Page for answering Question 4

Solution:

```
# arrays in my universe are indexed from 1 to n

sort(A)

for i = 1 to n:
        result = binary_search(A, B[i])
                # Assume binary_search returns -1 if not found
                # Some students will supply the first and last index
                #   values for A, such as "binary_search(A,1,n,B[i])"
        if result != -1:
                print B[i]
```

The complexity of the the sort is $O(n \log n)$. The loop executes n times and each iteration includes an $O(\log n)$ call to `binary_search` so the complexity of the loop is $O(n \log n)$ as well.

Thus the algorithm runs in $O(n \log n)$ time.

Marking:

Students are not required to state the complexity of their algorithm – I included that analysis to demonstrate that $O(n \log n)$ is possible.

| | |
|---|---|
| For a correct $O(n \log n)$ algorithm | 10 marks |
| For a correct $O(n^2)$ algorithm | 6 marks |
| For a correct algorithm of higher complexity | 5 marks |
| | |
| For a minor mistake (eg index error) | -2 |
| For a major mistake (eg fatal logic error) | -4 |

Notwithstanding the above, a student who clearly understood the question

and tried to answer it should get at least 4 marks.

**BONUS QUESTION ( |Ø| marks)**

What is the meaning of this figure?