Name: _____ **NSID:** _____ **Student #:** _____

University of Saskatchewan
Department of Computer Science
CMPT 145.3
Midterm Examination
February 15, 2017

Marks: 39 **Time: 50 minutes**

********** Solution & Marking Guide **********

NSID: _____

Part I — Multiple Choice

Choose the best answer for each question. Choose only one answer per question.

Section 1: Multiple Choice Questions

(1) 1. Suppose a program goes into an infinite loop. Which one of the design and implementation goals has not been met?

A. Correctness B. Efficiency C. Robustness D. Adaptability E. Reusability

(1) 2. Suppose a program crashes (terminates unexpectedly) when it cannot connect to the Internet. Which one of the design and implementation goals has not been met?

A. Correctness B. Efficiency **C. Robustness** D. Adaptability E. Reusability

(1) 3. Suppose a program had to be completely rewritten because of an update to the operating system (e.g., Windows, macOS, LINUX). Which one of the design and implementation goals has not been met?

A. Correctness B. Efficiency C. Robustness **D. Adaptability** E. Reusability

(1) 4. When we talk about algorithm efficiency, we are concerned with space and time.

A. TRUE B. FALSE

(1) 5. Which of the following statements about data types is the best description?
This question was poorly written and did not count against your midterm grade.

A. Only types predefined by Python, like integers and lists, are data types.

B. A data type is completely described by its data values alone.

C. A data type has two parts: data values and operations on its data values.

D. A data type has three parts: data values, operations on its data values, and a data structure.

E. A data type is completely described by comments in a Python program.

(1) 6. A data structure is a data type with compound data values that are organized in a particular way.

A. TRUE B. FALSE

(1) 7. Which one of the following statements about Abstract Data Types (ADTs) is false?

A. An ADT defines a way to encapsulate data.

B. An ADT defines the operations that can be performed on the data.

C. An ADT hides details about how the operations work behind its interface.

D. An ADT is defined by its implementation, not its interface.

E. An ADT protects data from unrestricted access.

(1) 8. In Lab 3, we experimented with the queueing simulation by replacing the FIFO queue for customers with a LIFO stack. Imagine instead that we took the bracket-matching program, and replaced the LIFO stack we used to contain `'('` with a FIFO queue. Which of the following best describes the consequences this replacement?

A. The program would simply not run at all.

B. Given a string with matching brackets, e.g., `'(())'`, the program would report an error.

C. Given a string with unmatched brackets, e.g., `'())'`, the program would fail to report an error.

D. The program would give the same answers as the original program.

E. None of the above.

(1) 9. Consider the following Python code, using the node ADT:

```
y = node.create(3,next=None)
node.set_next(y, node.create(4,next=None))
x = y
y = node.get_next(y)
node.set_next(y, node.create(5,next=None))
```

Which of the following diagrams correctly describes the node chain created above?

A. `x-->[ 5 | *-]-->[ 4 | *-]-->[ 3 | / ]`

B. `x-->[ 3 | *-]-->[ 4 | *-]-->[ 5 | / ]`

C. `x-->y-->[ 4 | *-]-->[ 5 | / ]`

D. `y-->[ 3 | *-]-->[ 4 | *-]-->[ 5 | / ]`

E. `y-->[ 5 | *-]-->[ 4 | *-]-->[ 3 | / ]`

(1) 10. Which of the following options correctly places the time complexity classes in order from slowest growing to most quickly growing, as n increases?
This question was did not test CMPT 145 material, and did not count against your midterm grade

A. $O(n^{1/2}) \leq O(\log n) \leq O(n) \leq O(n \log n) \leq O(n!)$
B. $O(\log n) \leq O(n^{1/2}) \leq O(n) \leq O(n \log n) \leq O(n!)$
C. $O(n^{1/2}) \leq O(\log n) \leq O(n^2) \leq O(n \log n) \leq O(n!)$
D. $O(n^{1/2}) \leq O(\log n) \leq O(n \log n) \leq O(n^2) \leq O(n!)$
E. $O(\log n) \leq O(n) \leq O(n^2) \leq O(n \log n) \leq O(n!)$

(1) 11. Express the following using Big-O notation $\boxed{10n + n^{10} + 10 \log n}$

A. $O(1)$ B. $O(n)$ **C.** $O(n^{10})$ D. $O(\log n)$ E. none of the above

(1) 12. Express the following using Big-O notation $\boxed{42 \log n + 5n^3 + \frac{1}{3} n \log n + \frac{n^4}{7}}$

A. $O(n^4)$ B. $O(n^3)$ C. $O(2^n)$ D. $O(\log n)$ E. $O(\sqrt{n})$

(1) 13. What is the worst case time complexity for the following Python fragment? Assume `print` is $O(1)$.

```
for i in range(n):
    print(i)
```

A. $O(n^2)$ **B.** $O(n)$ C. $O(2n)$ D. $O(n^3)$ E. $O(3^n)$

(1) 14. What is the worst case time complexity for the following Python fragment? Assume the function `work(...)` is $O(n^2)$ (and that the details about its interface are not important).

```
for i in range(n):
      work(...)
```

A. $O(n)$ B. $O(2n)$ C. $O(n^2)$ **D.** $O(n^3)$ E. $O(3^n)$

(1) 15. What is the worst case time complexity for the following Python fragment? Assume `print` is $O(1)$.

```
for i in range(n):
    for j in range(n):
        print(i,j)
```

A. $O(n)$ B. $O(2n)$ **C.** $O(n^2)$ D. $O(n^3)$ E. $O(3^n)$

(1) 16. What is the worst case time complexity for the following Python fragment? Assume `work(...)` is $O(n^2)$ (and that the details about its interface are not important).
This question had an error, and did not count against your midterm grade

```
i = 0 # should be 1!
while i < n:
    j = 0
    while j < n/2:
        work(...)
        j = j + 1
    i = i * 2
```

A. $O(n \log n)$ B. $O(n^3)$ **C.** $O(n^3 \log n)$ D. $O(n^4)$ E. $O(n^4 \log n)$

Part II — Written Answers.

Answer each question in the space provided on this question paper.

(10) 17. In this question you will demonstrate your understanding of the Queue and Stack ADTs. Use the ADT operations only.

Write a function named `reverse` with one parameter, a Queue, that returns a new Queue, with the same elements as the original queue, but with the order reversed.

You may assume that the Queue and Stack modules are already imported as Queue and Stack. Write a careful doc-string, documenting the interface for your function.

Hint: You must use the ADT operations. You may not access the Queue or Stack directly.

Solution:

```
def reverse(queue):
    """
    Purpose:
      Return a new queue whose elements are the same as the given queue
      but whose order is the reverse
    Pre-conditions:
      queue: an instance of the Queue ADT
    Post-conditions:
      -the data in the new queue is the reverse of the given queue
      -the original queue is empty
    Return:
      a new queue
    """
    s = Stack.create()
    q = Queue.create()
    while not Queue.is_empty(queue):
        Stack.push(s, Queue.dequeue(queue))
    while not Stack.is_empty(s):
        Queue.enqueue(q, Stack.pop(s))
    return q
```

Grading:

- 2 marks: Interface.
  - Deduction of 1 mark if the post-condition is incorrect or missing. Only the post-condition is at all interesting, because the obvious algorithm leaves the original queue empty.
  - Deduction of 2 marks if the interface is not documented at all.
- 4 marks: Algorithm: Using FIFO/LIFO together to reverse the items
  - Deduction of 2 marks if FIFO behaviour not used.
  - Deduction of 2 marks if LIFO behaviour not used.
  - Deduction of 2 marks if FIFO/LIFO coordination is incoherent
- 4 marks: Use of ADTs.
  - Deduction of 2 marks for violating the ADT, including treating Stacks and Queues as interchangeable
  - Deduction of 1 marks for violating the ADT, including len(), for-in, etc.
  - Deduction of 2 marks if ADT operations used incoherently.
  - No deduction for minor syntax problems in ADT use.

Notes:

- POST: You didn't document the post-condition, or your documentation left out the most important fact.
- DOC: You didn't write a doc-string at all, or the comments you wrote did not document the function.
- ADT vio(l): Your code violated the ADT for Stack or Queue or both.
- FIFO/LIFO: Something about your use of queues and stacks. Sometimes used to indicate something done correctly, sometimes used to annotate deductions.
- ADT incoh: Your use of the ADT was incoherent.
- ALG: Something wrong with your algorithm.

(13) 18. In this question you will demonstrate your understanding of the Node ADT.

(a) (6 pts) Write a code fragment in Python to count the number of nodes in a node-chain. You may assume that the variable `chain` refers to the first node in the node-chain.

Solution:

```
length = 0
anode = chain
while anode != None:
    anode = Node.get_next(anode)
    length += 1
```

Grading:

- Note: This question was taken out of 6, not 7 as on your exam paper.
- 1 mark for a counter
- 1 mark for a variable to step through the chain
- 1 mark for the while loop condition
- 1 mark for using the Node ADT operation get_next() correctly
- 1 mark for updating the position in the node-chain
- 1 mark for updating the counter

Notes:

- empty: You didn't handle the case for an empty chain.
- Don't move chain: Always use a separate variable for stepping through the chain.

(b) (7 pts) Write a code fragment in Python to insert a new node in the node-chain starting at `chain`. Assume that you know the length of the node chain, stored in the variable `length`. Put the new node half-way down the chain. For example, if the chain is 7 nodes long, put the new node after the 3rd node (i.e., use integer division by 2 to determine the appropriate place). The new node should store the value 42.

Solution:

```
newnode = node.create(42)
if length == 0:
    chain = newnode
else:
    anode = chain
    count = 0
    while count < length//2:
        anode = node.get_next(anode)
        count += 1
    node.set_next(newnode, node.get_next(anode))
    node.set_next(anode, newnode)
```

Grading:

- 1 mark for the new node
- 1 mark for checking if chain is empty, and attaching the new node correctly
- 2 marks for a loop to step to the middle of the chain, checking NULL and using next; this is part (a) except for the stopping criterion
- 2 marks for setting connecting the rest of the chain to newnode
- 1 mark for connecting anode to newnode correctly

Notes:

- Empty: you did not handle the case for an empty chain.
- Off-by-one: You put the new node in the wrong place.
- Connect: something was wrong with your attempt to connect the new node to the rest of the chain