UNIVERSITY OF TORONTO
Faculty of Arts & Science

Winter 2023 Midterm

CSC 148 H1S

Duration: 110 min.

Aids Allowed: None

*Do **not** turn this page until*
you have received the signal to start.
In the meantime, fill in your
name, student number, and UTORid,
and carefully read all the information below.

First (Given) Name(s):

Last (Family) Name(s):

10-Digit Student Number:

UTORid (e.g., `pitfra12`):

This test consists of 5 questions on 14 pages (including this one), printed on both sides of the paper.

There is a separate aid sheet provided. You do not need to hand it in, and nothing you write on it will be marked.

There are blank pages at the end of the test for rough work. **Do not remove them.**

You may always write helper functions unless asked not to.

Documentation is *not* required unless asked for.

You may write in either pen or pencil.

Good luck! We want you to do well!

Marking Guide

Nº 1: _____/10

Nº 2: _____/ 7

Nº 3: _____/ 9

Nº 4: _____/12

Nº 5: _____/ 5

TOTAL: _____/43

STUDENTS MUST HAND IN ALL EXAMINATION MATERIALS AT THE END

Winter 2023 Midterm
CSC 148 H1S
Duration: **110 min.**

Question 1. [10 marks]

Part (a) [1 mark]

Suppose we have a class called `Kettle`, with an initializer and one additional method: `boil`. Which of these is it appropriate for `Kettle`'s representation invariants to mention? Circle all that apply.

1. parameters of the initializer for class `Kettle`
2. private attributes of class `Kettle`
3. public attributes of class `Kettle`
4. parameters of method `boil`

Part (b) [1 mark]

Suppose class `High` is the parent of class `Low`. What methods does `Low` inherit from `High`? Circle one.

1. all methods in `High`
2. all methods in `High` except the initializer
3. all public methods in `High`
4. all methods in `High` that don't access private attributes
5. no methods from `High`

Part (c) [3 marks]

If we run this code, what is printed? Write your answer to the right of the code.

```
def cow(this: int, that: int) -> int:
    try:
        print('start cow')
        if this / that > 5:
            return 1
        else:
            return 2
    except ZeroDivisionError:
        print('problem with cow!')
        return -1

def lamb(junk: dict[int, int]) -> None:
    try:
        print('start lamb')
        value = cow(junk[1], junk[2])
        junk[value] = 42
    except KeyError:
        print('problem with lamb!')

if __name__ == '__main__':
    d = {1: 25, 2: 0, 3: 10}
    lamb(d)
    print(-1 in d)  # Prints either True or False
```

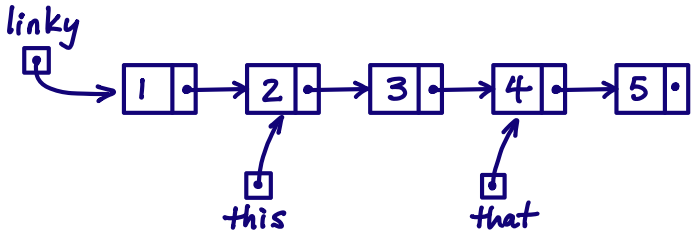
Part (d) [3 marks]

Complete the following hypothesis test so that it checks one aspect of `list.reverse`. (The aid sheet includes an example of what `list.reverse` does.) Note: `min_size=1` ensures that hypothesis only generates lists with at least 1 element.

```
@given(lists(integers(), min_size=1))
def test_reverse(lst: List):
    """Test that
    """
    # TODO: Complete the docstring description and implement this function.
```

Part (e) [2 marks]

Suppose we have set up variables `linky` (of type `LinkedList`), and `this` and `that` (of type `_Node`) as follows:



If we print `linky` we see the following:

```
[1 -> 2 -> 3 -> 4 -> 5]
```

Suppose we now run the line of code below:

```
this.next.next = that.next.next
```

It runs without error. What will we see if we print `linky` now?

Question 2. [7 marks]

In the code below, we are trying to model a Menu that records various Foods, including their prices and quantities. It isn't working properly.

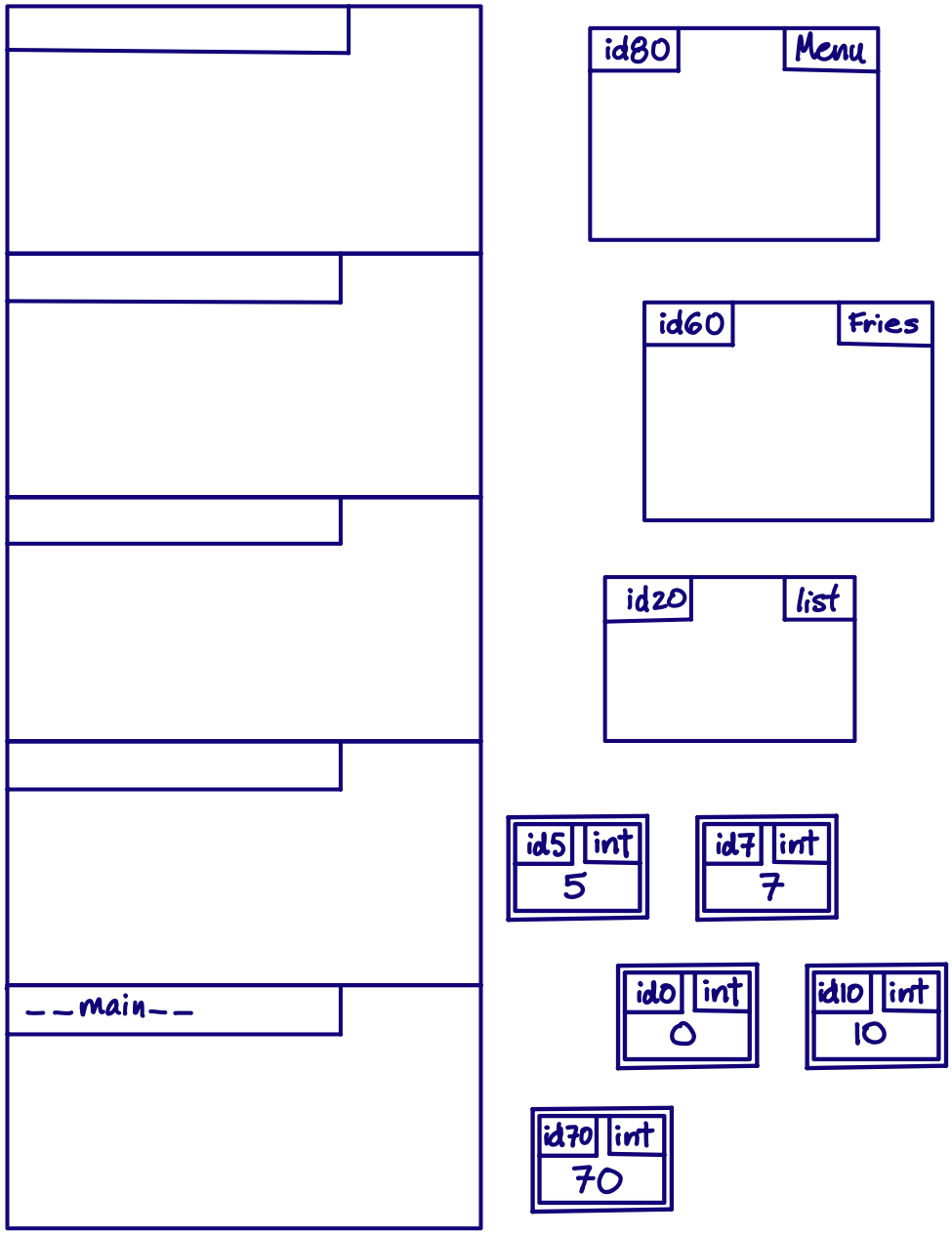
```
class Food:
    def __init__(self, p: int) -> None:
        self.price = p
        self.quantity = 0

    def restock(self, new_quantity: int) -> None:
        """Set this Food's quantity to new_quantity."""
        quantity = new_quantity


class Fries(Food):
    def restock(self, num_batches: int) -> None:
        new_quantity = num_batches * 10
        Food.restock(self, new_quantity)


class Menu:
    def __init__(self, items: list[Food]) -> None:
        self.items = items


if __name__ == '__main__':
    f = Fries(5)
    menu = Menu([f])
    f.restock(7)
    print('done')
```

There is a problem somewhere in the code which is preventing the quantity from updating properly. To locate the problem, we have begun drawing a memory model diagram that traces through the code in the `if __name__ == '__main__'` block.

Part (a) [5 marks]

On the next page, complete the diagram to show the state of memory immediately before the print statement in the main block. We have provided some empty stack frames and partially created some objects you may need.

You must show all intermediate steps. Cross out any stack frames that are removed from the call stack, and any values that are changed. When labelling the frame for a call to a method, include the class name; e.g., write `Food.restock` or `Fries.restock`.



Part (b) [1 mark]

Make the smallest change possible that will fix the code so that our quantity will update properly. Make your change directly to the code on the previous page.

Part (c) [1 mark]

In the initializer for Menu, we set `self.items = items`. Though this does not cause any problems with the code we've written, it may cause a problem in other client code. Explain *what* problem could arise and *why*.

Question 3. [9 marks]

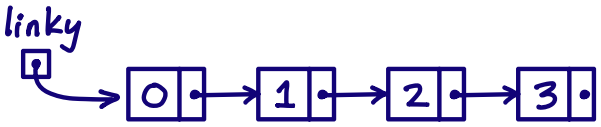
We are working on adding a new method for our `LinkedList` class, as shown below. The docstring is correct, however the body of the method has a bug.

```
    def insert_at(self, item: Any, pos: int) -> None:
        """Insert <item> at position <pos> in this list, or after the last node
        if <pos> is bigger than the length of the list.

        Precondition: pos >= 1
        """
 1      if self._first is None:
 2         self._first = _Node(item)
 3      else:
 4          curr = self._first
 5          i = 0
 6          while curr.next is not None and i != pos:
 7              curr = curr.next
 8              i += 1
 9          new_node = _Node(item)
10          new_node.next = curr.next
11          curr.next = new_node
```

Part (a) [1 mark]

The diagram below shows the linked list created using the code

```
linky = LinkedList([0, 1, 2, 3])
```



Use the space above to trace the following call to the method. If it runs without error, draw a new diagram below that shows the state of the linked list afterwards. If it raises an error, state what line number it occurs on and explain what sort of error it is (you don't have to know the precise error name).

```
linky.insert_at(99, 2)
```

Part (b) [3 marks]

In the table below, fill in what happens for the example call above, and then write three additional test cases for the new method. These must test scenarios that are distinct from each other and from the one test we have given. You must include at least one test that the code passes, and at least one that it fails. Write "P" or "F" in the Pass/Fail? column to indicate whether the method passes or fails the test (raising an error counts as failing the test).

Note: we represent our linked list as its values in the style of a Python list here for clarity and brevity.

Scenario	Initial list	Arguments to Method	Expected Result	Actual Result / Reason for Error Raised	Pass / Fail?
not specified	`[0, 1, 2, 3]`	`99, 2`	`[0, 1, 99, 2, 3]`		

Part (c) [3 marks]

Complete the `pytest` function below to implement the test case from our example method call in Part (a). We have given you the header. Note: you can use the `__eq__` method (as on the aid sheet) to compare two linked lists.

```
def test_example_call() -> None:
    """Test the insert_at method with the arguments from Part (a) of this question.
    """
```

Part (d) [2 marks]

The bug in the code can be fixed by rewriting **exactly one line of code** in the method. Indicate the line number that you think needs to be fixed, and then write the corrected line of code here. Rewrite only one line of the method.

Line number: ☐

Corrected line of code: _____

Question 4. [12 marks]

In this question, you will develop some code in a similar way to the SuperDuperManager worksheet, where you were given a specification and some client code.

Specification: There are many different possible kinds of Question. Every question has question text and a correct answer. A multiple choice question has a list of answer options, but a text question does not. Any question can check a potential answer for correctness. The difficulty of a multiple choice question is the number of answer options in it, while the difficulty of a text question is always 10. Given a list of questions, our code should be able to calculate the total difficulty of the questions.

Write code such that the main block below would execute successfully. To earn full credit you must also:

- make appropriate use of inheritance and polymorphism,
- include appropriate type annotations for parameters, return values, and instance attributes, and
- make your code consistent with the specification above.

Do not write any docstrings.

```
if __name__ == '__main__':
    # This question has the three answer options given, and the correct answer is Ottawa:
    q1 = MultipleChoice('What is the capital of Canada?',
                        ['Toronto', 'Ottawa', 'Vancouver'], 'Ottawa')
    print(q1.difficulty())                      # Prints 3
    # The answer for this question is Meric Gertler:
    q2 = TextQuestion('Who is the president of UofT?', 'Meric Gertler')
    print(q2.check_answer('Sophia'))            # Prints False
    quiz = [q1, q2]
    print(total_difficulty(quiz))               # Prints 13
```

Winter 2023 Midterm

CSC 148 H1S

Duration: **110 min.**

Continue your solution on this page if needed.

Question 5. [5 marks]

Consider the following `GridADT` which specifies some common operations for working with a two-dimensional n-by-n grid of values.

```
class GridADT:
    """An abstract class representing an n-by-n grid of values.
    === Public Attributes ===
    n: the number of rows and columns in the grid. Row indexes and
       column indexes go from 0 to n-1
    === Representation Invariants ===
    - n >= 1
    - If a location in the grid has not yet had its value set, the value None
      is associated with that location. See the get_value method.
    """
    n: int

    def __init__(self, size: int) -> None:
        self.n = size

    def get_row(self, row: int) -> list[Any]:
        """Return the n values stored in the given <row> of the grid.
        Precondition: <row> is a valid row index
        """
        raise NotImplementedError

    def get_column(self, col: int) -> list[Any]:
        """Return the n values stored in the given <col> of the grid.
        Precondition: <col> is a valid column index
        """
        raise NotImplementedError

    def get_value(self, row: int, col: int) -> Optional[Any]:
        """Return the value stored at the given <row> and <col> in the grid.
        If no value is yet associated with this location in the grid, None is
        returned.
        Precondition: <row> is a valid row index and <col> is a valid
            column index
        """
        raise NotImplementedError

    def set_value(self, row: int, col: int, value: Any) -> None:
        """Set the <value> at the given <row> and <col> in the grid.
        Precondition: <row> is a valid row index and <col> is a valid
            column index
        """
        raise NotImplementedError
```

Winter 2023 Midterm

CSC 148 H1S

Duration: **110 min.**

Part (a) [4 marks]

Implement the following function, which makes use of the `GridADT` class. Include appropriate precondition(s) for your function.

```
def set_row_values(grid: GridADT, row_index: int, row_values: list) -> None:
    """Set the values of <grid>, so that the row with index <row_index> contains
    the values stored in <row_values>, in the order that they appear in
    <row_values>.

    Precondition(s): (add your precondition(s) below)



    """
```

Part (b) [1 mark]

Assume the function from part (a) was implemented correctly. Suppose we wanted to add a doctest example to the docstring for the previous function. What would happen if we tried to run the following doctest? Explain why.

```
    >>> g = GridADT(5)
    >>> row_index = 0
    >>> row_values = [0, 1, 2, 3, 4]
    >>> set_row_values(g, row_index, row_values)
    >>> g.get_row(row_index)
    [0, 1, 2, 3, 4]
```

Use this page for rough work.
If you want the work on this page to be marked, indicate this clearly at the location of the original question.

Winter 2023 Midterm

CSC 148 H1S

Duration: **110 min.**

Use this page for rough work.
If you want the work on this page to be marked, indicate this clearly at the location of the original question.

Use this page for rough work.
If you want the work on this page to be marked, indicate this clearly at the location of the original question.