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 two additional methods: `fill` and `boil`. What code is responsible for ensuring that `Kettle`'s representation invariants hold? Circle all that apply.

1. the initializer for class `Kettle`
2. the `fill` method for class `Kettle`
3. the `boil` method for class `Kettle`
4. client code of the `Kettle` class

Part (b) [1 MARK]

What code can assume that the preconditions of `Kettle`'s `boil` method hold? Circle all that apply.

1. the initializer for class `Kettle`
2. the `boil` method for class `Kettle`
3. client code of the `Kettle` class

Part (c) [3 MARKS]

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

```
def horse(stuff: dict[int, int]) -> int:
    try:
        print('start horse')
        stuff[6] = 35
        return chicken(stuff)
    except KeyError:
        print('problem with horse!')
        return -1

def chicken(junk: dict[int, int]) -> int:
    try:
        print('start chicken')
        answer = junk[42]
        return answer
    except ZeroDivisionError:
        print('problem with chicken!')
        return 13

if __name__ == '__main__':
    d = {5: 25, 3: 9}
    answer = horse(d)
    print(6 in d)   # Prints either True or False
    print(answer)
```

start horse
start chiken
problem with horse
True
-1

WINTER 2023 MIDTERM

CSC 148 H1S

Duration: **110 min.**

Part (d) [3 MARKS]

Complete the following hypothesis test so that it checks one aspect of `list.clear`. (The aid sheet includes an example of what `list.clear` does.)

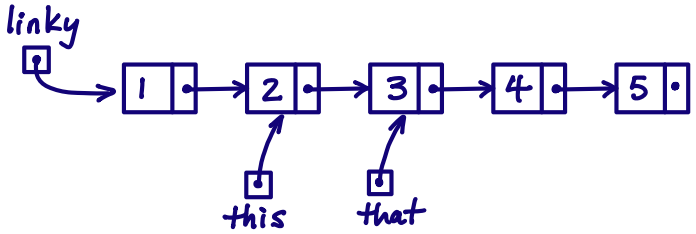
```
@given(lists(integers()))
def test_clear(lst: List):
    """Test that
    """
    # TODO: Complete the docstring description and implement this function.
```

Test that lists.clear() always returns lst == []

assert lst.clear() == []

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?

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

Question 2. [7 marks]

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

```
class Food:
    def __init__(self, n: str, p: int) -> None:
        self.name = n
        self.price = p

    def update_price(self, new_price: int) -> None:
        """Update this Food's price to new_price."""
        price = self.price
        price = new_price
```

self.price = new_price

```
class Fries(Food):
    def __init__(self, p: int) -> None:
        Food.__init__(self, 'fries', p)


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


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

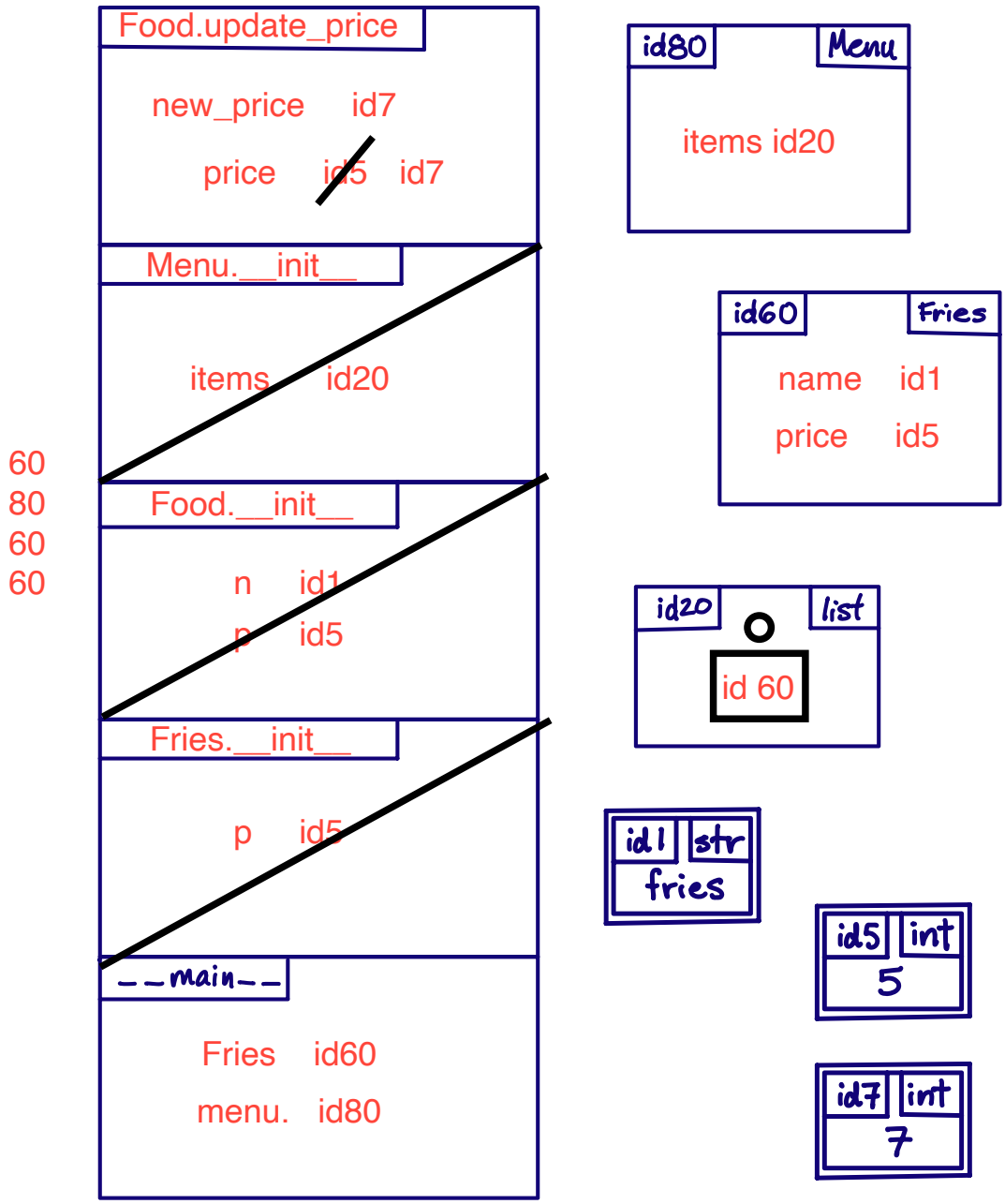
There is a problem somewhere in the code which is preventing the price from updating properly. To locate the problem, we have begun drawing a memory model diagram that traces through the code in the `if __name__ == '__main__'` block.

Part (a) [5 marks]

On the next page, complete the diagram to show the state of memory immediately before the call to `update_price` returns. 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 `Fries.__init__` or `Food.__init__`.

Tex



Part (b) [1 mark]

Make the smallest change possible that will fix the code so that our price 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*.

since items and Menu.items are two diffrent valuables

Question 3. [9 marks]

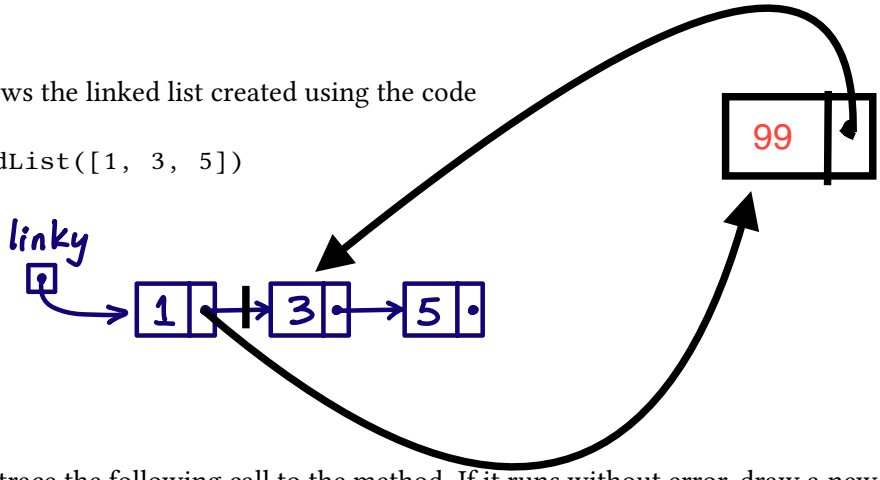
We are working on adding a new method for our `LinkedList` class, as shown below. The docstring is correct, however the body of the method has a bug.

```
    def insert_after(self, item: Any, after: Any) -> None:
        """Insert <item> into this list after the first occurrence of <after>
        or at the end of the list if <after> does not appear.
        """
1       if self._first is None:
2           self._first = _Node(item)
3       else:
4           curr = self._first
5           while curr.next is not None and curr.next.item != after:
6               curr = curr.next
7           new_node = _Node(item)
8           new_node.next = curr.next
9           curr.next = new_node
```

Part (a) [1 mark]

The diagram below shows the linked list created using the code

```
linky = LinkedList([1, 3, 5])
```



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_after(99, 3)
```

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	`[1, 3, 5]`	`99, 3`	`[1, 3, 99, 5]`	[1, 99, 3, 5]	F
empty list	[]	148, 165	[148]	[148]	P
last item(item exists)	[2, 4, 8]	16, 8	[2, 4, 8, 16]	[2, 4, 16, 8]	F

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_after method with the arguments from Part (a) of this question.
    """
```

```
linkey = LinkedList([1, 3, 5])
linkey.insert_after(99,3)

assert linkey == LinkedList([1, 3, 99, 5])
```

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: 5

Corrected line of code: while curr.next is not None and curr.item != after:

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: A Polygon is made of some number of straight sides. There are many different possible kinds of polygons. Every polygon can have its area calculated and has a given number of sides. A square is a polygon with four sides, each of equal length, so its area is the square of its side length. A right triangle is a polygon with three sides. Given the length of its base and its height, the area of a right triangle is given by one half the product of its base times height. Given a collection of polygons, our code should be able to calculate the total area of all of the provided polygons.

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__':
    s = Square(5.0)                    # Length of a side is 5.0
    print(s.sides)                     # Prints 4
    print(s.area())                    # Prints 25.0
    t = RightTriangle(2.0, 3.0)        # Base is 2.0 and height is 3.0
    print(total_area([s, t]))          # Prints 28.0
```

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 get_from_grid(grid: GridADT, cell_indexes: list[tuple[int, int]]) -> list[Any]:
    """Return a list of values from <grid> corresponding to each (row, col) pair
    in <cell_indexes>. For any (row, col) pair that has no value associated in
    <grid>, put a None in the list.

    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)
    >>> g.set_value(0, 0, 1)
    >>> g.set_value(4, 4, 16)
    >>> indices = [(0, 0), (4, 4)]
    >>> get_from_grid(g, indices)
    [1, 16]
```

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.