CSCA48 Winter 2017 Final Exam
Duration — 2 hours 50min
Aids allowed: none

Student Number: _____ **A**

Markus Login: _____

Last Name: _____ **First Name:** _____

Question 0. [1 MARK]

Carefully read and follow all instructions on this page, and fill in all fields.

---

*Do **not** turn this page until you have received the signal to start.*

This exam consists of 6 questions on 20 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Proper documentation is required for all functions and code blocks. If you use any space for rough work, indicate clearly what you want marked. Please read all questions thoroughly before starting on any work. Any pages detached from this cover page will not be marked.

The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University.

# 0:	_____/ 1
# 1:	_____/12
# 2:	_____/ 5
# 3:	_____/13
# 4:	_____/ 8
# 5:	_____/ 8
# 6:	_____/ 3
TOTAL:	_____/50

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 1. [12 MARKS]

For this question assume that L is a list of integers and its length is n.

For each of the following pseudocode segment, give its complexity using big-Oh notation. E.g., $O(3^n)$. *No justification is required.*

(a)
```
sum = 0
for i in range(len(L)):
    for j in range(i):
        sum = sum + L[i] * L[j]
```

(b)
```
sum = 0
for i in range(len(L)):
    j = 1
    while j < len(L):
        sum = sum + L[i] * L[j]
        j = j * 2
```

(c) For this part assume that *bst* is an initially empty binary search tree.

```
for i in range(len(L)):
    insert L[i] into bst
```

(d) For this part assume that *pq* is an initially empty min heap that uses a list to represent a compact binary tree as shown in class.

```
for i in range(len(L)):
    insert L[i] into pq
```

(e) For this part assume that *pq* is in the state immediately after the pseudocode segment from part (d) is executed.

```
i = 1
while i < len(L):
    extract minimum integer from pq
    i = i * 2
```

(f) For this part assume that *pq* is an initially empty min heap that uses a list to represent a compact binary tree as shown in class.

```
i = 1
while i < len(L):
    insert L[i] into pq
    i = i * 3
```

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 2. [5 MARKS]

Recall from your assignment the `whowins` function, which takes a linked list representation of a snakes and ladders board, plus step sizes for players 1 and 2, and returns the number of the player who wins the game on the given board with players using their given step sizes. Design a suite of test cases for `whowins`. For your convenience one example of a test case (with a rather poor explanation) is given.

number of squares	list of snadders	player 1 step size	player 2 step size	who wins	explanation
101	17 to 71, 73 to 37	2	5	2	cuz we love palindromic prime numbers

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 3. [13 MARKS]

Consider an operation, called *shuffle*, which reorders the items in a given list by swapping the first and second items, and the third and fourth items, and the fifth and sixth items, and so on until the end of the list. If there is an odd number of items in the list, then the last item remains at the end after a shuffle. Here are some examples.

- After a shuffle, the list $1, 2, 3, 4, 5, 6$ becomes $2, 1, 4, 3, 6, 5$.
- After a shuffle, the list $1, 2, 3, 4, 5$ becomes $2, 1, 4, 3, 5$.
- The empty list, or any list of one item, is unchanged by a shuffle.

Nick wrote a function that shuffles a doubly linked list using the following class for nodes.

```
class DLLnode:
    """ A node for a doubly linked list of objects."""
    def __init__(self: 'DLLnode', data: object,
                 prev: 'DLLnode', nxt: 'DLLnode') -> None:
        self.data = data
        self.prev = prev  # backward pointer to previous node in DLL
        self.nxt = nxt    # forward pointer to next node in DLL
```

Then the Code Stealer struck! All of Nick's pointers were stolen and replaced by "blanks" (strings of underscores). For example, a line like

```
head.nxt = None
```

would be gone and replaced by

```
---------- . ---------- = ----------
```

Please restore Nick's function by filling in each blank with one of the following:
`before`, `curr`, `prev`, `nxt`, `head`, `None`.

```
def shuffle(head: 'DLLnode') -> 'DLLnode':
    """
    Shuffle the doubly linked list whose first node is head.
    Return the head of the updated list.
    """
    # if list not empty

    if __________ != __________:

        # set before, curr to point at first, second node respectively

        __________ = __________

        __________ = __________ . __________

        # shuffle 2 nodes at a time until end of list

        while __________ != __________:

            # set what should point to curr

            if __________ . __________ != __________:

                __________ . __________ . __________ = __________
            else:

                __________ = __________

            # set forward pointers

            __________ . __________ = __________ . __________

            __________ . __________ = __________

            # loop continues on next page
```

```
        # fill in each blank with one of:
        #     before, curr, prev, nxt, head, None

        # set what should point back to before

        if __________ . __________ != __________:

            __________ . __________ . __________ = __________

        # set backward pointers

        __________ . __________ = __________ . __________

        __________ . __________ = __________

        # advance before and curr to next pair of nodes

        __________ = __________ . __________

        if __________ != __________:

            __________ = __________ . __________
        else:

            __________ = __________

    return __________
```

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 4. [8 MARKS]

Recall the `FormulaTree` class from the formula game assignment. Its subclasses include `AndTree`, `OrTree`, `NotTree` and `Leaf`. Here are its attributes.

`symbol` character stored at root (`*`, `+`, `-`, or a lower case letter).

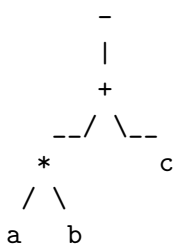
`children` list of pointers to subtrees of root, ordered from left to right.

Consider the following function which takes a `FormulaTree` and modifies it in some mysterious way.

```
def voodoo(root):
    if root.symbol == '-':
        voodoo(root.children[0])

    elif root.symbol in "*+":
        newchildren = []
        for c in root.children:
            voodoo(c)

            if c.symbol == root.symbol:
                newchildren = newchildren + c.children
            else:
                newchildren.append(c)

        root.children = newchildren
```

Part (a) [2 MARKS]

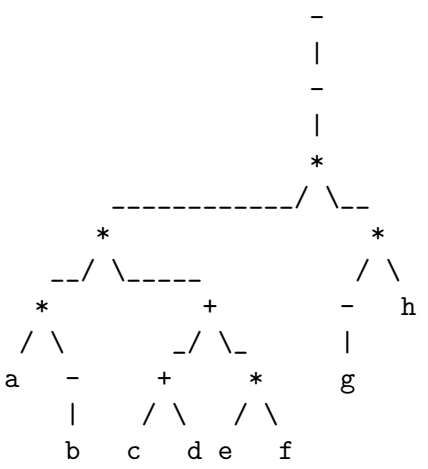
Draw the modified tree after executing `voodoo(root)`, where `root` is the root of the following tree.



[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Part (b) [2 MARKS]

Draw the modified tree after executing `voodoo(root)`, where `root` is the root of the following tree.



Part (c) [4 MARKS]

Give the function a better name and write a docstring for it.

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 5. [8 MARKS]

With the same `FormulaTree` class as in the previous question, Nick wrote a function `tree2formula(root)`, which takes the formula tree rooted at `root` and returns the formula represented by the tree.

Then disaster struck!

First the Comment Stealer erased all the internal comments.

Then the Code Mangler mangled all the lines of code by deleting all leading spaces and reordering the lines.

Here is how Nick's code looked afterwards.

```
def tree2formula(root: 'FormulaTree') -> str:
    """
    Return a string which is the formula represented by the FormulaTree
    rooted at root.

    (Essentially tree2formula is the "inverse" of build_tree.)
    """

elif root.symbol == '-':
else:
for c in root.children:
if 'a' <= root.symbol <= 'z':
result = '('
result = result[:-1] + ')'
result = result + tree2formula(c) + root.symbol
result = root.symbol
result = '-' + tree2formula(root.children[0])
return result
```

On the next page please restore Nick's code and internal comments.

For your convenience here is the mangled code again.

```
elif root.symbol == '-':
else:
for c in root.children:
if 'a' <= root.symbol <= 'z':
result = '('
result = result[:-1] + ')'
result = result + tree2formula(c) + root.symbol
result = root.symbol
result = '-' + tree2formula(root.children[0])
return result
```

```
def tree2formula(root: 'FormulaTree') -> str:
    """
    Return a string which is the formula represented by the FormulaTree
    rooted at root.

    (Essentially tree2formula is the "inverse" of build_tree.)
    """
```

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 6. [3 MARKS]

Thoughout this course we have seen many ADTs. Now it's time to vote for the 2017 Best A48 ADT Award. Queue and stack are not eligible. You cannot vote for them. Any ADT that was seen in a lecture, a tutorial, a practical, an exercise, an assignment, or a term test is eligible. Essentially any ADT that everyone in the class would have seen is eligible — except queue and stack, of course.

(a) Which ADT gets your vote?

(b) Give your reason(s) for your choice in part (a).

(c) Which ADT is your second choice?

Bonus. (CONTINUED)

Nick has been shopping around his new movie "Code Mangler: Origins" to various Hollywood studios. For this bonus question, you can either:

- Make a poster for the movie
- Write a good plot synopsis (brief overview of the story)
- Come up with a good marketing campaign
- Write a full script treatment, get a studio to option it, produce, direct, edit, and release the film (if it makes at least $100 million at the box office, you get double bonus points)
- Put something else in this space that the markers will find interesting/funny