CSCA48 Winter 2016 Final Exam
Duration — 180 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|

Markus Login: ______________________

Last Name: ______________________ **First Name:** ______________________

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

This exam consists of 7 questions on 22 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 can not fit your work in the space provided, you may use other rough work space, but you must clearly indicate where to find your work, and what work should be marked. Any pages not attached to this cover page will not be marked. Marks will be deducted for failing to properly fill out this cover page. Please read all questions thoroughly before starting on any work.

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.

# 1:	______/ 4
# 2:	______/ 4
# 3:	______/ 5
# 4:	______/ 8
# 5:	______/ 6
# 6:	______/11
# 7:	______/10
TOTAL:	______/48

[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. [4 MARKS]

For each of the following attempts at implementing a binary search, state whether or not the code is correct. If it is not, then give a smallest example of a list L and an intger x for which binsearch(L, x) would not return the correct result. By "smallest", we mean the length of L must be as small as possible.

```
def binsearch1(L, x):
    ''' (list of int, int) -> bool

    Return true iff x is in L.
    REQ: len(L) > 0 and L is sorted
    (in non-decreasing order).
    '''
    low = 0
    high = len(L)-1
    while (low < high):
        mid = (low + high) // 2
        if x <= L[mid]:
            high = mid
        else:
            low = mid
    return (x == L[low])
```

```
def binsearch2(L, x):
    ''' (list of int, int) -> bool

    Return true iff x is in L.
    REQ: len(L) > 0 and L is sorted
    (in non-decreasing order).
    '''
    low = 0
    high = len(L)
    while (high - low > 1):
        mid = (low + high) // 2
        if x <= L[mid]:
            high = mid
        else:
            low = mid + 1
    return (x == L[low])
```

[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. [4 MARKS]

Consider inserting the numbers in the set 1,2,3,4 into a binary search tree. Each order of insertion may yield a different tree. Give all the orders of insertion that yield a tree with maximum possible height.

Question 3. [5 MARKS]

Consider the following problem:

`Given n integers, find the third largest one.`

How efficiently this problem can be solved depends on how the n integers are stored. For each of the following way of storing integers, give the running time of the best algorithm to solve our problem. Use big-Oh notation.

Part (a) [1 MARK]

An unsorted list:

Part (b) [1 MARK]

A sorted list:

Part (c) [1 MARK]

A list representation of a min-heap:

Part (d) [1 MARK]

A list representation of a max-heap:

Part (e) [1 MARK]

A binary search tree:

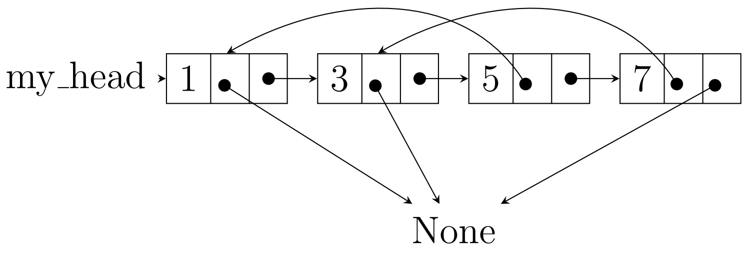
```
def f1b2insert(head, i):
    '''(F1B2Node, int) -> F1B2Node
    Insert a new F1B2Node with data i into sorted F1B2 list with
    first node head, maintaining sorted order of the list.
    REQ: F1B2 list with first node head is sorted.
    '''
    ________ = F1B2Node(i)

    # search for place to insert
    ________ = ________
    ________ = ________
    while (________ != ________ and i >= ________.data):
        ________.________ = ________
        ________ = ________
        ________ = ________.________

    # link new node to its next node
    ________.________ = ________

    # if inserting after first node of list
    if ________ != ________:
        ________.________ = ________
    else:
        ________ = ________

    # if inserting before last node of list
    if ________ != ________:
        ________.________ = ________

        # if inserting before second last node of list
        if ________.________ != ________:
            ________.________.________ = ________

    return ________
```

Question 4. [8 MARKS]

One Step Forward, Two Steps Back (F1B2)

Some students from the entirely fictional and not at all based on a real place Yark University decided to implement a linked list of integers where each node also contains an extra pointer to the node that is two places back in the list (if such a node exists). What a silly idea!

For example, here is a diagram of a list representing [1,3,5,7].



They want to use these F1B2 lists to hold sorted lists of integers. So they wrote a F1B2Node class.

```
class F1B2Node:
    '''
    A node for a F1B2 (forward one back two) list of integers.
    '''

    def __init__(self, i):
        '''(F1B2Node, int) -> NoneType
        Create a F1B2Node with data i.
        '''
        self.data = i
        self.back2 = None  # pointer to node 2 places back, if it exists
        self.forward = None  # pointer to next node if it exists
```

Then they got an A48 student from UTSC to help them write function to insert an integer into a sorted F1B2 list. But then when learning about regular expressions, they created an expression to replace
`(None | newnode | curr | prev | head | forward | back2)` with ______.

Your job is to restore the code on the previous page by filling in the blanks. All blanks must be filled with one of the following:

`None, newnode, curr, prev, head, forward, back2.`

```
from  ancient_tome import Stack
class StackQueue:
    '''
    A queue implemented with Stack objects.
    '''
    def __init__(self):
        '''(StackQueue) -> NoneType
        Create an empty StackQueue object.
        '''








    def enqueue(self, x):
        '''(StackQueue, object) -> NoneType
        Enqueue x into self.
        '''








    def dequeue(self):
        '''(StackQueue) -> object
        Dequeue and return the next object in self.
        REQ: self is not empty.
        '''








    def is_empty(self):
        '''(StackQueue) -> bool
        Return True iff self is empty.
        '''
```

Question 5. [6 marks]

The Isle of ADT consists of 5 countries: Stackland, Queueland, Listland, Dictionaryland, Setland. These countries are feuding and on verge of war. Things are so bad that King Stanley Von Stackton of Stackland decided to outlaw the use of queues, lists, dictionaries and sets. Stacklanders pleaded with him. "Our programs will stop working"! "Can we at least use queues"? But King Stanley was unmoved. As Chief Programmer of Stackland, you must implement a queue using only stacks. In the ancient library, you found some code written by the legendary ruler Nick the conqueror. But it seems that at some point during the great mangler wars of legend, the bodies of all of the methods were stolen!

The code that remains is on the previous page. You must complete all of the methods. You can use the sacred `stack` class, with its holy methods `push, pop, and is_empty`. You can not use any other data types (or you will have to answer to King Stanley).

1. Any string of length one containing a lower case letter is a condex.
2. If c is a condex, then so is '!' + c.
3. If c_1 and c_2 and c_3 are condexes, then so is: '(' + c_1 + '?' + c_2 + ':' + c_3 + ')'

```
mystery(s, i):
    # help with tracing:
    #           1         2
    # 012345678901234567890123456789
    # (!?:)!!c(o?!(n?d:e):!x)FUN!

    result = 0
    if 0 <= i < len(s):
        if 'a' <= s[i] <= 'z':
            result = i+1
        elif s[i] == '!':
            result = mystery(s, i+1)
        elif s[i] == '(':
            j = mystery(s, i+1)
            if 0 != j < len(s) and s[j] == '?':
                j = mystery(s, j+1)
                if 0 != j < len(s) and s[j] == ':':
                    j = mystery(s, j+1)
                    if 0 != j < len(s) and s[j] == ')':
                        result = j+1
    return result
```

Question 6. [11 MARKS]

We define a conditional expression (which we abbreviate as "condex") to be any string that can be constructed using the 3 rules on the previous page.

Some examples of condexes are: {`i, !o, (x?y:z), !(a?(!!b?!c:d):e)`}

Nick wants to write a function that determines whether a given string is a valid condex. He found an interesting function that may be helpful. Unfortunately whoever wrote it had some bad programming habits (obviously they never took A48). The variable names are terrible, as is the function name, and there is only one cryptic comment. The code Nick found can be seen on the previous page (and on the detachable sheet at the back of this exam)

Part (a) [2 MARKS]

Trace the following line of code, including what it returns:

```
mystery( "(!?:)!!c(o?!(n?d:e):!x)FUN!", 1)
```

[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) [5 MARKS]

Trace the following line of code, including what it returns.

```
mystery( "(!?:)!!c(o?!(n?d:e):!x)FUN!", 11)
```

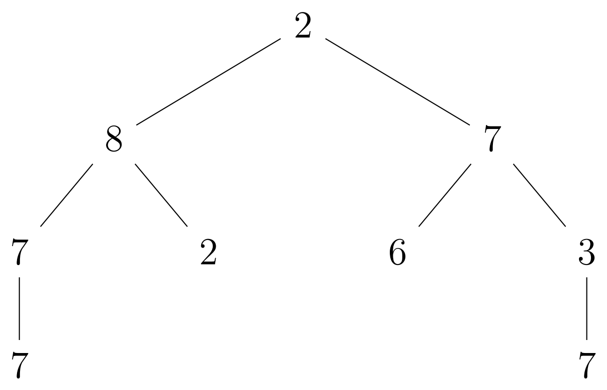
Part (c) [2 MARKS]

Give a better name and description for the function Nick found.

Part (d) [2 MARKS]

Use the function that Nick found to write a function called `is_condex` that takes a string s and returns whether s is a valid condex (No DocString necessary).

```
class MMNode:
    '''
    A node for a min-max
    '''

    def __init__(self, k, lchild=None, rchild=None):
        '''(MMNode, int, MMNode, MMNode) -> NoneType
        Create a new MMNode with key k, and children lchild and rchild.
        '''
        self.key = k
        self.left = lchild
        self.right = rchild
```



Question 7. [10 marks]

Consider a binary tree where each node contains an integer key. We say that such a tree is a min-max tree if it has the following properties

1. The key in any node at an even depth is less than or equal to the key in any of its children.
2. The key in any node at an odd depth is greater than or equal to the key in any of its children.

We can define an anti-min-max tree using the same definition except the words "even" and "odd" are interchanged. An example min-max tree, and the code for a min-max-tree node is given on the previous page. Note that the left and right subtrees of the root are anti-min-max trees.

Nick wrote a function, with a default/optional parameter, that determines whether the tree rooted at a given node is a min-max tree. Unfortunately, the CODE MANGLER struck.

- The docstring was erased.
- All lines of code were unindented and shuffled.
- Duplicate lines were removed.

The mangled code can be found on the next page (and on the detachable sheet at the back of this exam).

On the following page, please restore Nick's code, and write a new DocString.

```
def is_minmax(root, odd=False):
else:
if odd:
if result and root.right != None:
if root != None:
if root.left != None:
is_minmax(root.left, not odd))
is_minmax(root.right, not odd))
result = (root.key <= root.left.key and
result = (root.key <= root.right.key and
result = (root.key >= root.left.key and
result = (root.key >= root.right.key and
result = True
return result
```

```
def is_minmax(root, odd=False):
```

[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.]

Bonus. (CONTINUED)

If you were in charge of the CS curriculum at UTSC, what one thing would you change about the first year program?

Bonus. (CONTINUED)

5 bonus points to the first 5 students to stand up on their chair during the exam and shout out "O CAPTAIN MY CAPTAIN!"

Just kidding, the previous bonus question was a joke. You'll get nothing for standing on your chair except mocking laughs from the TAs and weird looks from students who haven't read the bonus questions yet. (Good thing you took the advice and read all the questions first before answering any of them, huh?)

Bonus. (CONTINUED)

For this bonus mark, you can either:

- Write a brief biography of the code mangler explaining why he/she is always ruining Nick's code
- Draw a picture that could be used on the poster for "CODE MANGLER: THE MOVIE"
- Write a poem about the code mangler

This page may be detached and used for code tracing or rough work. If detached, this page WILL NOT be marked

```
mystery(s, i):
    # help with tracing:
    #           1         2
    # 012345678901234567890123456789
    # (!?:)!!c(o?!(n?d:e):!x)FUN!

    result = 0
    if 0 <= i < len(s):
        if 'a' <= s[i] <= 'z':
            result = i+1
        elif s[i] == '!':
            result = mystery(s, i+1)
        elif s[i] == '(':
            j = mystery(s, i+1)
            if 0 != j < len(s) and s[j] == '?':
                j = mystery(s, j+1)
                if 0 != j < len(s) and s[j] == ':':
                    j = mystery(s, j+1)
                    if 0 != j < len(s) and s[j] == ')':
                        result = j+1
    return result
```

This page may be detached and used for code tracing or rough work. If detached, this page WILL NOT be marked

```
def is_minmax(root, odd=False):
else:
if odd:
if result and root.right != None:
if root != None:
if root.left != None:
is_minmax(root.left, not odd))
is_minmax(root.right, not odd))
result = (root.key <= root.left.key and
result = (root.key <= root.right.key and
result = (root.key >= root.left.key and
result = (root.key >= root.right.key and
result = True
return result
```