PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL 2012 EXAMINATIONS

CSC 148 H1S
Instructor(s): P. Gries / V. Pandeliev

Duration — 3 hours

Examination Aids: None

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

Family Name(s): _____

Given Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully.*

This final examination paper consists of 7 questions on 16 pages (including this one). *When you receive the signal to start, please make sure that your copy of the final examination is complete and fill in the identification section above.*

You don't have to write docstrings or comments except where we ask for them.

Unless stated otherwise, you are allowed to define helper methods and functions.

If you are unable to answer a question (or part), you will get 20% of the marks for that question (or part) if you write "I don't know" and nothing else. You will *not* get those marks if your answer is completely blank, or if it contains contradictory statements (such as "I don't know" followed or preceded by parts of a solution that have not been crossed off).

MARKING GUIDE

1: _____/12

2: _____/12

3: _____/10

4: _____/ 8

5: _____/10

6: _____/ 6

7: _____/10

TOTAL: _____/68

Good Luck!

CONT'D...

Question 1. [12 MARKS]

This question is about doubly-linked lists. Here is a node definition:

```
class Node(object):
    def __init__(self, o):
        self.data = o
        self.prev = None
        self.next = None
```

Part (a) [7 MARKS]

Complete function cut. **Strong hint:** this does not involve loops or recursion, and you'll almost certainly need to draw some pictures. Why not do Part (b) first?

```
def cut(front, tail, cutpoint):
    '''(Node, Node, Node) -> (Node, Node)
    front and tail are the first and last Nodes in a doubly-linked list. cutpoint is a
    Node somewhere in the list. Move all the Nodes before cutpoint to the end of the
    list (keeping their order the same), making cutpoint the new front. Return a tuple
    containing the first and last Nodes in the resulting doubly-linked list.
    Precondition: the list is not empty.
    Example: if the list contains the data 1, 2, 3, 4 and cutpoint points to the node
    containing 3, then the resulting list should contain the data 3, 4, 1, 2.'''
```

Part (b) [5 MARKS]

Think of a set of thorough test cases for `cut`. Draw one doubly-linked list per test case. In each picture, indicate which nodes are `front`, `tail`, and `cutpoint`. You do not need to write any code for this subquestion.

Question 2. [12 MARKS]

You are tasked to provide a generic framework for a card game that uses a standard deck of cards. A standard deck of cards has 52 cards, which come in four flavours, called *suits*: Spades, Hearts, Diamonds, Clubs. Every card also has one of thirteen values, which are called *ranks*: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King.

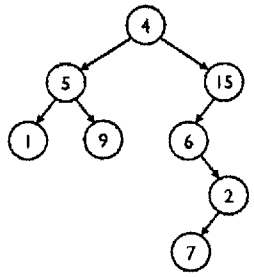
Developers will use your framework to design card games. The framework must provide common functionality such as shuffling, dealing a card (taking a card from the top of the deck), and adding card(s) to the deck.

For each class you would like to create, write the class definition, full `__init__` method, and headers and docstrings for all other methods. You *do not* need to write the code for any of the methods besides `__init__`.

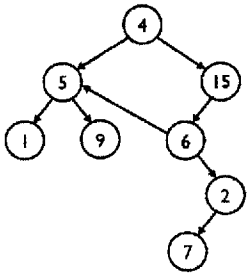
Continue your framework here.

Question 3. [10 MARKS]

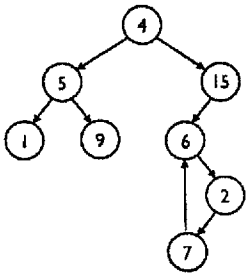
We call any structure in which nodes are connected by edges a *graph*. A tree is a type of graph that has no cycles in it: each node in a tree has at most one parent node. What this means is that no node is ever pointed to by more than one other node and, in any traversal of the tree, no node is ever seen more than once.



Tree



not a Tree



not a Tree

Given the following definition of class `Node`, complete the recursive function `is_tree(n)` (on the next page) that returns whether the graph rooted at `n` is in fact a binary tree (whether each node is encountered exactly once in a traversal). You may (and probably should) use a helper function.

```
class Node(object):
    def __init__(self, k):
        self.key = k
        self.left = None
        self.right = None
```

```
def is_tree(n):
    '''(Node) -> bool
    Return whether the graph rooted at n is a binary tree. An empty graph is a
    tree.'''
```

Question 4. [8 MARKS]

In class, we have defined a binary tree `Node` class with three attributes: `key`, `left` and `right`. In this question, we will consider a `Node` implementation that has a `parent` attribute as well. For the root `Node`, the `parent` attribute will be `None`. For every other `Node` in a binary tree, the `parent` attribute will the `Node`'s parent `Node`.

Consider the `Node` class defined below.

```
class Node(object):
    def __init__(self, k):
        self.key = k
        self.left = None
        self.right = None
        self.parent = None
```

Complete the function `in_same_tree` according to its docstring.

```
def in_same_tree(node1, node2):
    '''(Node, Node) -> bool
    Return whether node1 and node2 are in the same binary tree.'''
```

Question 5. [10 MARKS]

Recall that with PyGame you can draw a coloured rectangle outline on a `Surface` using this function:

```
pygame.draw.rect(screen, col, (x, y, width, height), 1)
```

Write a recursive function `draw_squares` that takes the following parameters:

- `screen`, a `Surface` to draw on,
- a tuple of 3 ints indicating the colour to draw with,
- ints `x` and `y` indicating a coordinate,
- an int `size` indicating the width and height of a square, and
- an int `size_diff`

Your function should draw concentric (nested) squares, each one `size_diff` pixels smaller than its enclosing square. What's a good base case?

Question 6. [6 MARKS]

Part (a) [2 MARKS]

What kind of data causes the worst case scenario for `quicksort`? Assume that the quicksort algorithm uses the first element of the list as the pivot.

Part (b) [2 MARKS]

Selection sort and insertion sort are both $O(n^2)$ sorts where n is the number of items being sorted. Which one performs fewer comparisons in its best case and why?

Part (c) [2 MARKS]

Consider the following two data structures:

- Linked List: A singly linked list with head and tail pointers
- Python List: A standard Python list

Circle the data structure that would be faster (in the worst case) for each of the following operations (assuming a very long list):

(i) inserting an element in a position near the beginning:	Linked List	Python List
(ii) inserting an element in a position near the end:	Linked List	Python List
(iii) deleting the last element:	Linked List	Python List
(iv) deleting the first element:	Linked List	Python List
(v) finding the middle element:	Linked List	Python List

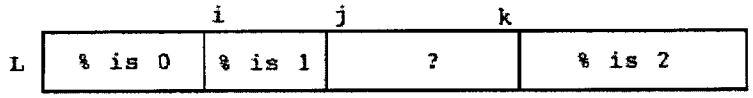
Question 7. [10 MARKS]

Recall that the modulo operator (`a % b`) returns the remainder of the integer division `a / b`.

Given a list of positive integers, we want to rearrange the list such that all the integers that give a remainder 0 when divided by 3 appear before those that give a remainder of 1, which appear before the ones that give remainder 2.

Example:
Before: `[3, 4, 1, 2, 6, 5]`
After: `[3, 6, 1, 4, 5, 2]`

Here is a hint:



At the beginning, all areas except the third are empty.

Write the function `modsort(L)` which partitions list L in-place. **No extra lists or other collections are allowed.** Note that the order of integers within each area does not matter: they do *not* need to be sorted.

For full marks, your solution should be $O(len(L))$.

```
def modsort(L):
    '''(list) -> NoneType
    Rearrange list L into three partitions that give remainder 0, 1 and 2
    when divided by 3.'''
```

Use this page for rough work and for answers that didn't fit. Indicate clearly what you want us to mark.

Use this page for rough work and for answers that didn't fit. Indicate clearly what you want us to mark.

YOU CAN TEAR THESE PAGES OFF IF YOU LIKE.

Short Python function/method descriptions:

```
__builtins__:
  len(x) -> integer
    Return the length of the list, tuple, dict, or string x.
  max(L) -> value
    Return the largest value in L.
  min(L) -> value
    Return the smallest value in L.
  open(name[, mode]) -> file object
    Open a file. Legal modes are "r" (read), "w" (write), and "a"
    (append).
  range([start], stop, [step]) -> list of integers
    Return a list containing the integers starting with start and
    ending with stop - 1 with step specifying the amount to increment
    (or decrement). If start is not specified, the list starts at 0.
    If step is not specified, the values are incremented by 1.
  raw_input([prompt]) -> string
    Read a string from standard input. The trailing newline is
    stripped.
  sum(L) -> number
    Returns the sum of the numbers in L.

dict:
  D[k] -> value
    Return the value associated with the key k in D.
  k in d -> boolean
    Return True if k is a key in D and False otherwise.
  D.get(k) -> value
    Return D[k] if k in D, otherwise return None.
  D.keys() -> list of keys
    Return the keys of D.
  D.values() -> list of values
    Return the values associated with the keys of D.
  D.items() -> list of (key, value) pairs
    Return the (key, value) pairs of D, as 2-tuples.

file (also called a "reader"):
  F.close()
    Close the file.
  F.read([size]) -> string
    Read and return at most size bytes. If the size argument is
    negative or omitted, read until EOF (End of File) is reached.
  F.readline([size]) -> string
    Return the next line from the file. Retain newline. A non-negative
    size argument limits the maximum number of b
    incomplete line may be returned then). Retur
    EOF.

float:
  float(x) -> floating point number
    Convert a string or number to a floating poi
    possible.

int:
  int(x) -> integer
    Convert a string or number to an integer, if
    point argument will be truncated towards zer

list:
  x in L -> boolean
    Return True if x is in L and False otherwise
  L.append(x)
    Append x to the end of list L.
  L1.extend(L2)
    Append the items in list L2 to the end of li
  L.index(value) -> integer
    Return the lowest index of value in L.
  L.insert(index, x)
    Insert x at position index.
  L.pop()
    Remove and return the last item from L.
  L.remove(value)
    Remove the first occurrence of value from L.
  L.reverse()
    Reverse *IN PLACE*
  L.sort()
    Sort the list in ascending order.

Module random:
  randint(a, b)
    Return random integer in range [a, b], inclu

str:
  x in s -> boolean
    Return True if x is in s and False otherwise
  str(x) -> string
    Convert an object into its string representa
  S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurre
```

CONT'D...

```
    in string S[start:end]. Optional arguments start and end are
    interpreted as in slice notation.
  S.find(sub[,i]) -> integer
    Return the lowest index in S (starting at S[i], if i is given)
    where the string sub is found or -1 if sub does not occur in S.
  S.isdigit() -> boolean
    Return True if all characters in S are digits and False otherwise.
  S.lower() -> string
    Return a copy of the string S converted to lowercase.
  S.lstrip([chars]) -> string
    Return a copy of the string S with leading whitespace removed. If
    chars is given and not None, remove characters in chars instead.
  S.replace(old, new) -> string
    Return a copy of string S with all occurrences of the string old
    replaced with the string new.
  S.rstrip([chars]) -> string
    Return a copy of the string S with trailing whitespace removed. If
    chars is given and not None, remove characters in chars instead.
  S.split([sep]) -> list of strings
    Return a list of the words in S, using string sep as the separator
    and any whitespace string if sep is not specified.
  S.strip() -> string
    Return a copy of S with leading and trailing whitespace removed.
  S.upper() -> string
    Return a copy of the string S converted to uppercase.

unittest.TestCase methods:
  assertTrue(self, expr[, msg])
    Check that the expression is true.
  assertFalse(self, expr[, msg])
    Check that the expression is false.
  assertEqual(self, first, second[, msg])
    Fail if the two objects are unequal as dete
    operator.
  fail(self[, msg])
    Fail immediately.

PyGame:
  pygame.display.set_mode((width, height)) -> S
    Return a Surface that is width pixels acros
  pygame.event.poll() -> Event
    Return an event that has happened, such as
    one of QUIT, MOUSEMOTION, KEYDOWN, MOUSEBUT
    KEYDOWN, KEYUP
  screen.fill(colour) -> NoneType
    Fill the screen Surface with a colour (a tu
  pygame.display.flip() -> NoneType
    Show the Surface on the computer screen.
  pygame.draw.rect(screen, col, (x, y, width, h
    Draw a filled rectangle width pixels wide a
    location (x, y) on the screen Surface using
    border is specified, draw a rectangle outli
    instead.
  pygame.font.Font(None, size) -> Font
    Return a Font object with font size use for
  font.render(text, antialias, col) -> Surface
    Return a new Surface containing str text in
    always use 1 for antialias.
  screen.blit(surface, (x, y)) -> NoneType
    Draw surface on the screen at position (x,
```

CONT'D...

Total Marks = 68