UNIVERSITY OF TORONTO

Winter 2011 Midterm

CSC148/A48: Introduction to Computer Science

Duration: 1 hour

March 2nd, 2011

Last Name: _____

First Name: _____

Student Number: _____

Instructions:

- **Write your name on the back of this exam paper.**
- **Do not open this exam until you hear the signal to start.**
- Have your student ID on your desk.
- No aids permitted other than writing tools. Keep all bags and notes far from your desk before the exam begins.
- There are 3 questions on 10 pages. When you hear the signal to start, make sure that your exam is complete before you begin.
- Read over the entire exam before starting.
- Write comments where it would clarify your code.
- If you use any space for rough work, clearly indicate the section(s) that you want marked.

Mark Breakdown

Question 1:	**/15**
Question 2:	**/15**
Question 3:	**/15**
Total:	**/ 45**

Question 1

Consider the following implementation of the Stack ADT, and then answer the question on the opposite page:

```
class EmptyStackError(Exception):
    pass

class Stack(object):
    """An implementation of the Stack ADT."""

    def __init__(self):
        """Create an empty Stack."""

        self._content = []

    def is_empty(self):
        """Return whether this Stack is empty."""

        return self._content == []

    def top(self):
        """Return the top item from this Stack. Raise EmptyStackError
        if the Stack is empty."""

        if self.is_empty():
            raise EmptyStackError, "top() called on empty Stack"

        return self._content[-1]

    def pop(self):
        """Remove and return the top item from this Stack. Raise
        EmptyStackError if the Stack is empty."""

        if self.is_empty():
            raise EmptyStackError, "pop() called on empty Stack"

        return self._content.pop()

    def push(self, item):
        """Add a new item to the top of this Stack."""

        self._content.append(item)
```

(a) In the space below, implement the `BoundedStack` class, where `BoundedStack` is a `Stack` with a limited capacity. If `push( )` is called on a full `BoundedStack`, it should raise a `FullStackError`.

(b) In the table below, describe the cases your PyUnit tester should use to fully test the `push( )` methods in both `Stack` and `BoundedStack`. An example has been provided below as an illustration.

Class to test	Initial values of attributes	Test call	Expected result
`Stack`	`_content = ['eh']`	`push('bee')`	`_content = ['eh', 'bee']`

Question 2

Consider the following operations. For each operation:

- write the worst-case time complexity (using big-oh notation) for a sensible algorithm
  e.g. $O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$, $O(n!)$, etc.
- describe the circumstances that would cause this worst-case performance

Keep your descriptions brief (one sentence) and provide the tightest bound possible.

Operation	Complexity	Explanation
Finding an element in an unsorted linked list with N nodes.		
Finding an element in a sorted Python list with N elements.		
Finding the smallest element in a binary tree with N nodes.		
Performing selection sort on a Python list that's already sorted.		
Checking if a number N with M digits is even or odd.		
Removing the 5^{th} element from a Python list of N elements (N>5).		
Finding all possible numbers that are N digits long.		
Adjusting the colour values of each Pixel in a Picture with height N and width M.		

Question 3

Implement the following `list_count` function so that it satisfies the specifications below:

```
def list_count(o):
   '''Return the number of lists in o.

      Parameters:
      o --- Python object

      Return:
      0 if o is not a list
      1 if o is an empty list
      1 + the number of lists in o's contents, otherwise

      Assumption:
      o is not infinitely nested.
   '''
```

Short Python function/method descriptions:

```
__builtins__:
    abs(x) -> number
      Return the absolute value of x.
    len(x) -> integer
      Return the length of the list, tuple, dict, or string x.
    max(L) -> value
      With a single iterable argument, return its largest item.
      With two or more arguments, return the largest argument.
    min(L) -> value
      With a single iterable argument, return its smallest item.
      With two or more arguments, return the smallest argument.
    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.
dict:
    D[k] or D.get(k) -> value
      Return the value associated with the key k in D.
    k in D or D.has_key(k) -> boolean
      Return True if k is a key in D and False otherwise.
    D.keys() -> list of keys
      Return the keys of D.
    D.values() -> list of values
      Return the values associated with the keys of D.
file (also called a "reader"):
    F.close()
      Close the file.
    F.read([size]) -> read at most size bytes, returned as a string.
      If the size argument is negative or omitted, read until EOF (End
      of File) is reached.
    F.readline([size]) -> next line from file, as a string. Retain newline.
      A non-negative size argument limits the maximum number of bytes to
      return (an incomplete line may be returned then). Return an empty
      string at EOF.
float:
    float(x) -> floating point number
      Convert a string or number to a floating point number, if possible.
int:
    int(x) -> integer
      Convert a string or number to an integer, if possible. A floating
      point argument will be truncated towards zero.
```

```
list:
   L.append(x)
     Append x to the end of the list L.
   L.index(value) -> integer
     Returns the lowest index of value in L.
   L.insert(index, x)
     Insert x at position index.
   L.remove(value)
     Removes the first occurrence of value from L.
   L.sort()
     Sorts the list in ascending order.
str:
   str(x) -> string
     Return a nice string representation of the object, if possible.
   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.index(sub) -> integer
     Like find but raises an exception if sub does not occur in S.
   S.isdigit() -> boolean
     Return True if all characters in S are digits and False otherwise.
   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
Exceptions:
   IndexException: occurs when sequence index is out of range.
```

This page is left blank intentionally for answer overflows.