CSC 148H5 S 2014 Test 1
Duration — 50 minutes
Aids allowed: none

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

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

Lecture Section: L0101 Instructor: Dan

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This test consists of 4 questions on 8 pages (including this page). *When you receive the signal to start, please make sure that your copy is complete.*
Comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code. No error checking is required: assume all user input and all argument values are valid.
If you use any space for rough work, indicate clearly what you want marked.

1: _____/ 6

2: _____/ 5

3: _____/ 5

4: _____/ 4

TOTAL: _____/20

Question 1. [6 MARKS]

Write the following function that operates on a stack. Assume that the `push`, `pop`, and `is_empty` stack methods are available. Do not call any method besides these three.

```
class StackException(Exception):
  pass


def swap_top(s: Stack) -> None:
  '''Swap the top two elements of Stack s.
  If there are fewer than two items in s,
  the stack is unchanged and a StackException is raised.

  >>> s = Stack()
  >>> s.push(1)
  >>> s.push(2)
  >>> swap_top(s)
  >>> s.pop()
  1
  '''
```

Question 2. [5 MARKS]

Here is a recursive function and a call of that function. What is the value of the call `rec(2, 2)`?

Please carefully show your work.

```
def rec(a: int, b: int) -> int:
  if a + b <= 0:
    return 1
  return rec(a - 1, b) + rec(a - 1, b - 1)

print(rec(2, 2))
```

Question 3. [5 MARKS]

A **palindrome** is a string that is the same forward and backward. As two different examples, `abba` and `racecar` are palindromes. The empty string is also a palindrome.

Write the following function. You must use recursion. Using a loop or helper function is not allowed.

```
def is_palindrome(s: str) -> bool:
  '''Return True iff s is a palindrome.
  '''
```

Question 4. [4 MARKS]

Here is a binary tree in list of lists format.

```
['a',
 ['b', ['c', ['d', None, None], None], None],
 ['e', None, ['f', None, ['g', ['h', None, None], None]]]
]
```

Give the order that the nodes are visited in an **inorder** traversal of this tree. Please show your work on your way to your response!

[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.]

[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.]

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

Short Python function/method descriptions:

```
__builtins__:
  input([prompt]) -> str
    Read a string from standard input; return that string with no newline. The prompt string,
    if given, is printed without a trailing newline before reading.
  max(a, b, c, ...) -> value
    With two or more arguments, return the largest argument.
  min(a, b, c, ...) -> value
    With two or more arguments, return the smallest argument.
  print(value, ..., sep=' ', end='\n') -> NoneType
    Prints the values. Optional keyword arguments:
      sep:  string inserted between values, default a space.
      end:  string appended after the last value, default a newline.
int:
  int(x) -> int
    Convert a string or number to an integer, if possible.  A floating point argument
    will be truncated towards zero.
str:
  S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.
  S.find(sub[,i]) -> int
    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.isalpha() -> bool
    Return True if and only if all characters in S are alphabetic
    and there is at least one character in S.
  S.isdigit() -> bool
    Return True if and only if all characters in S are digits
    and there is at least one character in S.
  S.islower() -> bool
    Return True if and only if all cased characters in S are lowercase
    and there is at least one cased character in S.
  S.isupper() -> bool
    Return True if and only if all cased characters in S are uppercase
    and there is at least one cased character in S.
  S.lower() -> str
    Return a copy of S converted to lowercase.
  S.replace(old, new) -> str
    Return a copy of string S with all occurrences of the string old replaced
    with the string new.
  S.split([sep]) -> list of str
    Return a list of the words in S, using string sep as the separator and
    any whitespace string if sep is not specified.
  S.startswith(prefix) -> bool
    Return True if S starts with the specified prefix and False otherwise.
  S.strip() -> str
    Return a copy of S with leading and trailing whitespace removed.
  S.upper() -> str
    Return a copy of S converted to uppercase.
```