CSC 148H5 S 2018 Test 1
Duration — 50 minutes
Aids allowed: none

Student Number: ⎵⎵⎵⎵⎵⎵⎵⎵⎵⎵

Last Name: ____________________ **First Name:** ____________________

☐	Lecture Section: L0101	Instructor: Larry Zhang (9:00-10:00)
☐	Lecture Section: L0102	Instructor: Larry Zhang (10:00-11:00)
☐	Lecture Section: L0103	Instructor: Dan Zingaro (13:00-14:00)
☐	Lecture Section: L0104	Instructor: Vincent Maccio (17:00-18:00)

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 10 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.
If you use any space for rough work, indicate clearly what you want marked.

1: ______/ 5

2: ______/ 5

3: ______/ 5

4: ______/ 5

TOTAL: ______/20

Question 1. [5 marks]

Recall what you did in Exercise 1, Question 1. Given a list of length `n` containing a permutation of the integers from `1` to `n`, you implemented a function that returns the maximum number of elements of the list (starting from `1`) that can be output in sorted order by using one auxiliary stack.

Part (a) [3 marks]

For each of the following lists, write down the correct return value of this function (a single integer). No justification is needed.

```
[2, 1, 4, 3]
```

```
[5, 6, 1, 3, 2, 4]
```

```
[8, 5, 1, 7, 2, 4, 6, 3]
```

Part (b) [2 marks]

Below, give a list of length 3 for which the correct return value would be 1. No justification is needed.

Question 2. [5 marks]

On the right side of this page, write down the output of the following program. Write the word “CRASH” at the end of your answer if the program terminates with an exception that is not handled.

```
class A:
    def __init__(self, a):
        self.num = a

    def double(self):
        self.num = self.num * 2

    def change(self):
        self.num = self.num + 1

class B(A):
    def __init__(self, a):
        A.__init__(self, a)

    def triple(self):
        self.num = self.num * 3

    def change(self):
        self.num = self.num - 1

if __name__ == "__main__":

    a = A(3)
    b = B(3)
    try:
        b.double()
        print("b =", b.num)
        a.triple()
        print("a =", a.num)
    except Exception:
        print("except")
    else:
        print("else")
        a.change()
    finally:
        print("finally")
        b.change()
    print("final a =", a.num)
    print("final b =", b.num)
```

```
WRITE THE OUTPUT BELOW HERE:
```

Question 3. [5 marks]

Write the below function which, given an input string `s`, returns a copy of `s` with each occurrence of `ab` substituted by `xy`. **Your code must be recursive** and you must **not** use any loop or any string/list/dict/etc. methods (e.g. `.replace`, `.find`, `.index`).

```
def replace_ab(s):
    '''(str) -> str
    Return a copy of s with each occurrence of 'ab' substituted by 'xy'

    >>> replace_ab('aab')
    'axy'
    >>> replace_ab('abcabc')
    'xycxyc'
    '''
```

Question 4. [5 marks]

Write the below function which, given a `Queue` object `q`, returns the number of elements stored in the queue. After the execution of the function, `q` **must be exactly the same as before**, i.e., it must contain the same elements in the same order.

In your implementation, you are allowed to use **only** one `Stack` object, i.e., you must **not** use Python list, set, dictionary, string, tuple, etc. On the `Queue` object, you're allowed to use the `enqueue`, `dequeue` and `is_empty` methods; on the `Stack` object, you're allowed to use `push`, `pop` and `is_empty`. **No other methods are allowed**.

```
def queue_size(q):
    '''(Queue) -> int
    Return the number of elements stored in q.
    After the execution of the function, q must be exactly the same as
    before, i.e., same elements in the same order.

    >>> q = Queue()
    >>> q.enqueue(7)
    >>> q.enqueue(8)
    >>> queue_size(q)
    2
    >>> q.dequeue()
    7
    >>> q.dequeue()
    8
    '''
```

[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.]

[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.
  isinstance(object, class) -> bool
    Return whether an object is an instance of a class or of a subclass thereof.
  print(value, ..., sep=' ', end='\n') -> NoneType
    Print 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.
list:
  L.append(object) -> None -- Append object to end.
  L.insert(index, object) -> None -- Insert object before index.
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.
```