CSC 148H5 S 2016 Test 1
Duration — 50 minutes
Aids allowed: none

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

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

☐	Lecture Section: L0102	Instructor: Dan Zingaro (11:00-12:00)
☐	Lecture Section: L0101	Instructor: Tiffany Tong (10:00-11:00)
☐	Lecture Section: L0103	Instructor: Sadia Sharmin (9:00-10:00)

*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 3 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.
If you use any space for rough work, indicate clearly what you want marked.

1: ______/ 6

2: ______/ 6

# 3: ______/10

TOTAL: ______/22

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.

```
def swap_bottom(s: Stack) -> None:
  '''Precondition: s has at least two elements.

  Swap the bottom two elements of Stack s.

  >>> s = Stack()
  >>> s.push(1)
  >>> s.push(2)
  >>> s.push(3)
  >>> swap_bottom(s)
  >>> s.pop()
  3
  >>> s.pop()
  1
  '''
```

Question 2. [6 marks]

Write the following function so that it satisfies its docstring. Your code *must be recursive.*

```
def duplicates(s: str) -> bool:
  '''
  Return True iff string s contains at least two adjacent characters that are the same.

  >>> duplicates('abab')
  False
  >>> duplicates('bbaa')
  True
  >>> duplicates('bba')
  True
  '''
```

Question 3. [10 marks]

Part (a)

Write a series of classes that satisfy the following specification.

- A To Do List has a name (an arbitrary string), and zero or more tasks, provided when the list is created.
- New tasks can be added to the To Do List, but the total number of tasks in the list must be 50 or less. Attempting to have more than 50 tasks in a list should cause a `TaskOverloadError`
- Each task has a day, month, and year, provided when the task is created. The day must be an integer between 1 and 31, the month must be an integer between 1 and 12, and the year must be greater than or equal to 2016. An invalid day, month or year should cause an `InvalidDateError`
- A task also has details about itself in a string (e.g. "date with Jane") provided when the task is created.

Part (b)

Write code to perform the following:

- Create a To Do List named "School" with an empty list of tasks
- Prompt the user for a day
  - *reminder*: use `input`
- Prompt the user for a month
- Create a task with the detail `Ace CSC148 test` using the given day and month, and the year 2016. Add this task to the To Do List that was created earlier.
  - If this raises an `InvalidDateError`, print `invalid date`
  - If this raises a `TaskOverloadError`, print `too many tasks`

[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.
  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.
```