CSC 108H1 F 2012 Midterm Test
Duration — 50 minutes
Aids allowed: none

Student Number: ______________________

Last Name: ___________________________ First Name: ___________________________

Lecture Section: L0101 L0102

Instructors (CIRCLE ONE): Michelle Craig Jennifer Campbell

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 midterm consists of 3 questions on 6 pages (including this one). 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: _____/ 4
# 2: _____/ 6
# 3: _____/10

TOTAL: _____/20

Total Pages = 6
Question 1. [4 marks]

Part (a) [1 mark] Write the output of the code below in the box below it.

```python
x = 4
y = x + 5
x = 7
print(x, y)
```

Part (b) [1 mark] Write the output of the code below in the box below it.

```python
s = 'coMPuTer'
i = 0
while i < len(s) and not s[i].isupper():
    print(s[i])
i = i + 1
```

Part (c) [1 mark] Fill in the box with Python code that will make the program behaviour match the comments. You may **not** make any other changes to the code or add code outside the box.

```python
def can_enter(height, with_adult):
    ''' (float, bool) -> bool
    Return True if and only if the child’s height is greater than 0.92 or
    the child is with an adult as indicated by with_adult.'''
    return
```

Part (d) [1 mark] Fill in the box with Python code that will make the program behaviour match the comments. You may **not** make any other changes to the code or add code outside the box.

```python
s1 = "midterm"
s2 = "science"

# Using only s1, s2, concatenation, and indexing and/or slicing, print the string 'test'.
print( )
```
**Question 2.** [6 marks]

**Part (a) [4 marks]**

For the function below, complete the function header (using a meaningful function name) and write a docstring (including the type contract, description and two examples).

```python
def
result = ''
for ch in s:
    if ch.isdigit():
        result = result + ch
    else:
        result = result + '-'
return result
```

**Part (b) [2 marks]**

Write a Python program that prompts the user to enter a string, calls the function from Part (a) passing the string that the user entered as an argument, and prints the value returned by the function. You may not change the function.
Question 3.  [10 Marks]

Part (a)  [5 Marks] Complete the following function according to its docstring.

HIDDEN = '^'

def merge_views(view1, view2):
    ''' (str, str) -> str
    view1 and view2 are views of the same puzzle. Some of the alphabetic characters may be revealed and some may be hidden (that is, replaced by HIDDEN). Return a view of the puzzle where the revealed characters are those that are revealed in at least one of view1 and view2.
    >>> merge_views('^~~^l^', 'a^^^e')
    'a^^le'
    >>> merge_views('^ppl^', '^pp^e')
    '^pple'
    '''

>>> merge_views('^~~^l^', 'a~~e')
'a~~le'
>>> merge_views('^ppl^', '^pp^e')
'^pple'

}
Part (b) [5 MARKS] The estimated cooking time for potatoes is usually 5 minutes per potato, but if the potatoes are extra large, it is 7 minutes per potato. When cooking more than 2 potatoes of any size, the cooking time calculated is too long, so 3 minutes is deducted from the estimated cooking time. (All potatoes being cooked together are the same size.)

Complete the following function according to the description above and the docstring below.

```python
def cooking_time(num_potatoes, extra_large):
    ''' (int, bool) -> int

    Return the estimated time in minutes to cook num_potatoes potatoes depending on whether or not they are extra_large.

    >>> cooking_time(5, True) # 5 extra large potatoes
    32
    >>> cooking_time(6, False) # 6 regular potatoes
    27
    >>> cooking_time(2, False) # 2 regular potatoes
    10
    '''
```

```python
>>> cooking_time(5, True) # 5 extra large potatoes
32
>>> cooking_time(6, False) # 6 regular potatoes
27
>>> cooking_time(2, False) # 2 regular potatoes
10
```
Short Python function/method descriptions:

`__builtins__`:
- `input([prompt]) -> str`
  
  Read a string from standard input. The trailing newline is stripped. 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='
') --> 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.