CSC A08 2013 Midterm Test
Duration — 50 minutes
Aids allowed: none

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

Instructor: Brian Harrington

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

Please place a checkmark (✓) beside your tutorial session

Tutorial Number	Date/Time	TA Name	Check
TUT0001	WE11:00 - 13:00	Denning Campbell	
TUT0002	WE11:00 - 13:00	Shichu Lin	
TUT0003	FR09:00 - 11:00	Ekin Ozcelik	
TUT0004	WE15:00 - 17:00	Nick (Ruo Fan) Li	
TUT0005	WE15:00 - 17:00	Nicholas Olson-Harris	
TUT0006	FR13:00 - 15:00	Philip (Jianhao) Yang	
TUT0007	TH11:00 - 13:00	Umair Idris	
TUT0008	TH11:00 - 13:00	Kenneth Ma	
TUT0009	TH13:00 - 15:00	Eric Ren	
TUT0010	TH13:00 - 15:00	Edouard Magharian	
TUT0011	TH15:00 - 17:00	Cyan Kuo	
TUT0012	TH15:00 - 17:00	Harmen (Hardarshan) Kahlon	
TUT0013	FR09:00 - 11:00	Faisal Usmani	
TUT0014	MO10:00 - 12:00	Gabrielle Singh-Cadieux	
TUT0015	MO10:00 - 12:00	Yamn Chalich	
TUT0016	MO10:00 - 12:00	Judy Duong	
TUT0017	TU13:00 - 15:00	Michelle (Xiaopeng) Cui	

---

*Do **not** turn this page until you have received the signal to start.*
Please fill out the identification section
and read all instructions before starting.
Good Luck!

This midterm consists of 4 questions on 12 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.* Proper documentation is required for all functions and code blocks. 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. Please read all questions thoroughly before starting on any work.

1: _____/ 5
2: _____/ 5
3: _____/10
4: _____/20

TOTAL: _____/40

[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.]

Question 1. [5 MARKS]

Write the output of the following code in the space provided

```
my_string = "Welcome to CSCA08!"
result = ""
for next_letter in my_string:
    if (next_letter.isalpha()):
        result += "X"
    elif (next_letter.isdigit()):
        result += "#"
    else:
        result += "!"
    if (next_letter in "AEIOU"):
        print("VOWEL")
    if (next_letter in "BRIAN"):
        print("Hi Brian")
print(result)
```

[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.]

Question 2. [5 MARKS]

Write the output of the following code in the space provided

```
my_list = ["1", 1, "2", 2, "3", 3, "4", 4]
i = 0
res = ""
while (i < len(my_list) and (isinstance(my_list[i], str) or (int(my_list[i]) < 3))):
    res += str(my_list[i])
    print(res)
    i += 1
```

[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.]

Question 3. [10 MARKS]

Write the output of the following code in the space provided

```
def my_func(input_var):
    input_var = 3

def my_func2(x):
    x = 3
    return x

def my_func3(input_list):
    my_list = input_list[:]
    my_list[0] = 9
    return my_list

def my_func4(input_list):
    input_list[0] = 9
    return "Hello"

x = 8
y = 8
print("STEP 1:", x)
print("STEP 2:", x + y)
z = x
x = 10
print("STEP 3:", z)
y = my_func(x)
print("STEP 4:", x, " - ", y)
x = 10
y = my_func2(x)
print("STEP 5:", x, " - ", y)
x = [1, 2, 3, 4]
y = my_func3(x)
print("STEP 6:", x, " - ", y)
x = [1, 2, 3, 4]
y = my_func4(x)
print("STEP 7:", x, " - ", y)
```

[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.]

Question 4. [20 MARKS]

Write a function called `cap_mask` that takes two strings as input. The first is a normal string, and the second is a mask containing 0s and 1s. Apply the mask to the string, and return the result.

That is: The returned string will have the same characters as the input string, except that: If the i^{th} character of the mask is `"0"`, the i^{th} character of the returned string will be `lower case`. If the i^{th} character of the mask is `"1"`, the i^{th} character of the returned string will be `upper case`. Characters in the input string which are not letters, will be left unchanged regardless of the value of the mask.

Hints:

- You must follow the design recipe
- If you can"t get something to work, write comments explaining what you WANT to do, you may receive part marks.
- As long as the type contract is fulfilled, your code shouldn"t crash. If the REQ statements are ignored, it doesn"t have to return anything sensible.

[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.]

Short Python function/method descriptions:

You may tear this page off, but if you do so, you must not include any work on it (front or back) that you wish to have marked.

```
__builtins__:
  abs(number) -> number
    Return the absolute value of the given number.
  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.
  isinstance(object, class-or-type-or-tuple) -> bool
    Return whether an object is an instance of a class or of a subclass thereof.
    With a type as second argument, return whether that is the object's type.
  int(x) -> int
    Convert a string or number to an integer, if possible.  A floating point argument
    will be truncated towards zero.
  str(x) -> str
    Convert an object into a string representation.

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 uppercase
    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.
```

```
list:
  append(...)
    L.append(object) -- append object to end
  count(...)
    L.count(value) -> integer -- return number of occurrences of value
  index(...)
    L.index(value, [start, [stop]]) -> integer -- return first index of value.
    Raises ValueError if the value is not present.
  insert(...)
    L.insert(index, object) -- insert object before index
  pop(...)
    L.pop([index]) -> item -- remove and return item at index (default last).
    Raises IndexError if list is empty or index is out of range.
  remove(...)
    L.remove(value) -- remove first occurrence of value.
    Raises ValueError if the value is not present.
```