CSCA08 Fall 2015 Term Test #2

Duration — 110 minutes
Aids allowed: none

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

Markus Login: ________________________

Last Name: ________________________ **First Name:** ________________________

Please place a checkmark (✓) beside your tutorial session

Tutorial Number	Date/Time	Room	TA Name	Check
TUT0001	WE 16:00 17:00	IC 212	Umair Idris	
TUT0002	WE 19:00 20:00	MW 264	Eric Wang	
TUT0003	MO 10:00 11:00	PO 101	Shadman Shadid	
TUT0004	TU 9:00 10:00	AA 209	Maheshan Indralingam	
TUT0005	TH 17:00 18:00	IC 212	Andrew Wang	
TUT0006	TH 18:00 19:00	BV 361	Ekin Ozcelik	
TUT0007	TH 19:00 20:00	BV 361	Kalindu De Costa	
TUT0008	TU 12:00 13:00	MW 140	Shichong Peng	
TUT0009	FR 9:00 10:00	IC 208	Yasaman Mahdaviyeh	
TUT0010	FR 10:00 11:00	MW 120	Ayaan Chaudhry	
TUT0011	FR 13:00 14:00	BV 355	Umair Idris	
TUT0012	FR 14:00 15:00	IC 212	Ekin Ozcelik	
TUT0013	TU 9:00 10:00	IC 230	Vidhya Arulnathan	
TUT0014	TU 11:00 12:00	IC 326	David Kua	
TUT0015	TH 12:00 13:00	MW 160	Pat McGee	
TUT0016	TU 9:00 10:00	HW 215	Bo Zhao	
TUT0017	TH 18:00 19:00	HW 308	Ben Cooper	
TUT0018	FR 9:00 10:00	IC 120	Mohammed Faizan	
TUT0019	FR 10:00 11:00	BV 363	Tianxiang Gao	
TUT0021	FR 11:00 12:00	IC 230	Judy Duong	
TUT0022	WE 9:00 10:00	BV 260	Pat McGee	
TUT0023	MO 9:00 10:00	AA 208	Janarthanan Manoharan	
TUT0025	WE 15:00 16:00	IC 120	Judy Duong	
TUT0026	FR 14:00 15:00	BV 355	Charles Ruan	

*Do **not** turn this page until you have received the signal to start.*

This exam consists of 3 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. If you use any space for rough work, indicate clearly what you want marked. Any pages not attached to this cover page will not be marked. Please read all questions thoroughly before starting on any work.

The University of Toronto's Code of Behaviour on Academic Matters applies to all University of Toronto Scarborough students. The Code prohibits all forms of academic dishonesty including, but not limited to, cheating, plagiarism, and the use of unauthorized aids. Students violating the Code may be subject to penalties up to and including suspension or expulsion from the University.

# 1:	______/ 5
# 2:	______/10
# 3:	______/15
TOTAL:	______/30

[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.

```
string1 = "ABCDE"
string2 = "12345"
for i in range(len(string1) - 1, -1, -1):
    count = 0
    res = ""
    while(count < len(string2)):
        if(i < count):
            res += string1[i]
        elif(i == count):
            res += "X"
        else:
            res += string2[count]
        count += 1
    print(res)
```

[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. [10 marks]

Write the output of the following code in the space provided.

```
def func1(l1):
    l1[0] = "X"
    print(l1)

def func2(l1, l2):
    l1[0] = l2[0]
    l3 = l1[0:3]
    l3[0] = "Y"
    l2[0] = l3[0]
    print(l1, l2, l3)

def func3(l1, l2):
    l3 = l1[:]
    l1[0] = l2
    l1[0][0] = "Z"
    l2[1] = "W"
    print(l1, l2, l3)

l1 = [1, 2, 3]
l2 = [7, 8, 9]
func1(l1)
func2(l1,l2)
l1 = [1, 2, 3]
l2 = [7, 8, 9]
func3(l1, l2)
l1 = [[1, 2], 3]
l2 = [[7], [8, [9]]]
func3(l1, l2)
```

MANGLED CODE:

```
count += 1
count = 1
def pig_latin(input_string):
else:
else:
first_vowel_index = count
for next_line in test_file:
for next_word in next_line_words:
found_vowel = False
found_vowel = True
if(__name__ == "__main__"):
if(input_string[0] in "aeiou"):
if(input_string[count] in "aeiou"):
next_line_words = next_line.split()
print(result)
result += "ay"
result += pig_latin(next_word) + " "
result = ""
result = input_string + "w"
result = input_string[first_vowel_index:] + input_string[0:first_vowel_index]
return result
test_file = open("test_file.txt", "r")
test_file.close()
while(count < len(input_string) and not found_vowel):
```

RULES OF PIG LATIN

- Words beginning with a vowel, have 'way' put on the end ('one' becomes 'oneway')
- Words with no vowels, have 'ay' put on the end ('xkcd' becomes 'xkcday')
- All other words have all of the letters before the first vowel moved to the end, and 'ay' put on the end ('brian' becomes 'ianbray')
- The code only works with lower case letters

Question 3. [15 marks]

Nick wrote this cool function that translates words into pig latin, and he wrote some global code to test it out, by translating every word in a file into pig latin. However, when he wasn't looking, the CODE MANGLER messed up all his code. He deleted all the comments, removed all indentation, and scrambled the lines. Your task is to re-construct Nick's code from the jumbled mess left by the code mangler (on the previous page). The rules of pig latin are also available on the previous page.

[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.]

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 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.
```

```
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.

math:
   ceil(...)
        Return the ceiling of x as an int.
        This is the smallest integral value >= x.
    cos(...)
        Return the cosine of x (measured in radians).
    floor(...)
         Return the floor of x as an int.
        This is the largest integral value <= x.
    pow(...)
        Return x**y (x to the power of y).
    sin(...)
        Return the sine of x (measured in radians).
    sqrt(...)
        Return the square root of x.
    tan(...)
        Return the tangent of x (measured in radians).
set:
  pop(...)
    Remove and return an arbitrary set element.
    Raises KeyError if the set is empty.
dict:
  keys(...)
    D.keys() -> a set-like object containing all of D's keys
  get(...)
     D.get(k[,d]) -> returns D[k] if k is in D, otherwise returns d.  d defaults to None.
object:
  __init__(...)
    x.__init__(...) initializes x; called automatically when a new object is created
  __str__(...)
    x.__str__() <==> str(x)
other:
   x // y = integer divide x by y (i.e., how many times does x divide evenly into y). 5 // 3 = 1
   x % y = the remainder when x is integer divided by y. 5 % 3 = 2
```