

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science
APRIL 2012 EXAMINATIONS

PLEASE HAND IN

CSC 108 H1S
Instructors: Campbell

Duration — 3 hours

Examination Aids: None

Student Number: _____

Family Name(s): _____

Given Name(s): _____

*Do **not** turn this page until you have received the signal to start.*
*In the meantime, please read the instructions below **carefully**.*

This final examination paper consists of 11 questions on 20 pages (including this one). *When you receive the signal to start, please make sure that your copy of the final examination is complete.*

Comments and docstrings 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.

You do not need to put `import` statements in your answers.

You may not use `break` or `continue` on this exam.

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

Assume all input is valid unless otherwise indicated; there is no need to error-check.

1: _____/ 4

2: _____/ 4

3: _____/ 4

4: _____/ 8

5: _____/ 8

6: _____/ 4

7: _____/ 8

8: _____/12

9: _____/12

10: _____/ 8

11: _____/ 4

TOTAL: _____/76

Question 1. [4 MARKS]

For each block of code in this question, write its output in the box below it. If it would generate an error, say so, and give the reason for the error.

Part (a) [1 MARK]

```
D = {}  
D['apple'] = 4  
D['apple'] = 8
```

```
print D
```

'apple' : 8

Part (b) [1 MARK]

```
D = {2 : ['d'], 0 : ['c']}  
D[0].append('a')  
print D[0]
```

Part (c) [1 MARK]

```
D = {4 : 3, 3 : 2, 2 : 1}  
for x in D.values():  
    print D[x]
```

Part (d) [1 MARK]

```
D1 = {'a' : 1, 'b' : 2, 'c' : 3}  
D2 = {'b' : 2, 'c' : 3, 'a' : 1}  
print D1 == D2
```

Question 2. [4 MARKS]

For each block of code in this question, write its output in the box below it. If it would generate an error, say so, and give the reason for the error.

Part (a) [1 MARK]

```
L1 = ['yesterday', 'today']
L2 = L1.append('tomorrow')
print L2
```

Part (b) [1 MARK]

```
L = [['on', 'off'], \
      ['go', 'slow', 'stop'], \
      ['up']]
print L[1][2][3]
```

Part (c) [1 MARK]

```
s1 = 'hello'
s2 = s1
s2 = s2 + '-bye'
print s1
```

Part (d) [1 MARK]

```
L1 = ['hello']
L2 = L1
L2[0] = 'bye'
print L1
```

Question 3. [4 MARKS]

Part (a) [3 MARKS]

Write a good docstring for this function.

```
def mystery(a, b):
    '''

    '''

    i = 0
    while b != "" and i < len(a):
        if a[i] in b:
            b = b.replace(a[i], "")
            i += 1

    return b == ""
```

Part (b) [1 MARK]

Provide sample arguments for a call to function `mystery(a, b)` and give the value returned.

Value of a	Value of b	Return value

Question 4. [8 MARKS]

Complete the following functions according to their docstring descriptions.

Part (a) [4 MARKS] **Do not use the str method swapcase.**

```
def our_swapcase(s):  
    '''(str) -> str  
    Return a copy of the string s with uppercase letters converted to  
    lowercase, and lowercase letters converted to uppercase.'''
```

Part (b) [4 MARKS]

```
def length_dict(word_list):  
    '''(list of strs) -> dict of {int : list of strs}  
    Return a dictionary in which each key is a word length and each value is a  
    list of the words from word_list with that length. Only the lengths of the  
    words that appear in word_list should be included as keys in the dictionary.'''
```

Question 5. [8 MARKS]

The `media` module has a function called `media.copy` that takes a `Picture` and returns a new `Picture` that is a copy of the original. Implement your own version of `media.copy` without using `media.copy`. You will be marked for efficiency, as well as for correctness.

```
def our_copy(pic):  
    '''(Picture) -> Picture  
    Return a new picture that is a copy of pic.'''
```

Question 6. [4 MARKS]

Recall the following function from Exercise 3:

```
def get_letter_counts(letter_dict):  
    '''(dict of {str : list of strs}) -> dict of {str : int}  
    In letter_dict, each key is a single lowercase letter and each value is a  
    list of lowercase words that start with that letter. Based on letter_dict,  
    return a new dictionary in which each key is a single lowercase letter and  
    each value is the number of lowercase words that start with that letter.'''
```

Suppose that we want to test `get_letter_counts`. Describe four test cases that each test different “categories” of inputs. To describe each test case, give the `dict` that you would pass to `get_letter_counts`, the return value you expect, and the purpose of the test case. Do **not** write any code. We have given you one test case as an example; add four more.

Value of <code>letter_dict</code>	Return value	Purpose of this test case
<code>{}</code>	<code>{}</code>	empty dictionary

Question 7. [8 MARKS]

A “address file” contains one street address per line, such as:

```
100 Main Street
23 Spring Park Road
2012 Sunny Lane
4 Martin Luther King Drive
```

An “address list” is a list of lists of strs, where each inner list is a 3-element list of street number, name, and type. For example, the “address list” for the file above is:

```
[[ '100', 'Main', 'Street'], [ '23', 'Spring Park', 'Road'], \
[ '2012', 'Sunny', 'Lane'], [ '4', 'Martin Luther King', 'Drive']]
```

In the file, a single space separates the number, name and type. The format of each address is:

- the street number is an integer
- the street name is one or more words (with words separated by a single space)
- the street type is one word

Complete the following function according to its docstring description.

```
def load_addresses(numbers_file):
    '''(file) -> list of lists of strs
    Return an "address list" based on the contents of the given "address file".'''
```


Question 8. [12 MARKS]

In this question, you will implement part of a word search game. A word search board is a rectangular board of lowercase letters. For example:

r	b	t	x	m	k
e	a	t	x	d	s
x	b	a	t	o	p
q	y	v	x	d	l

A “word search file” contains a word search board. For example:

```
rbtxmk
eatxds
xbatop
qyvxdl
```

A “word search list” (a list of strs) represents a word search board. For example:

```
['rbtxmk', 'eatxds', 'xbatop', 'qyvxdl']
```

The object of this game is to identify words (sequences of characters) that appear in the board. A word can appear from top to bottom in a column, for example ***baby*** or ***spl***:

r	b	t	x	m	k
e	a	t	x	d	s
x	b	a	t	o	p
q	y	v	x	d	l

A word can also appear from left to right in a row, for example ***eat*** or ***vx***:

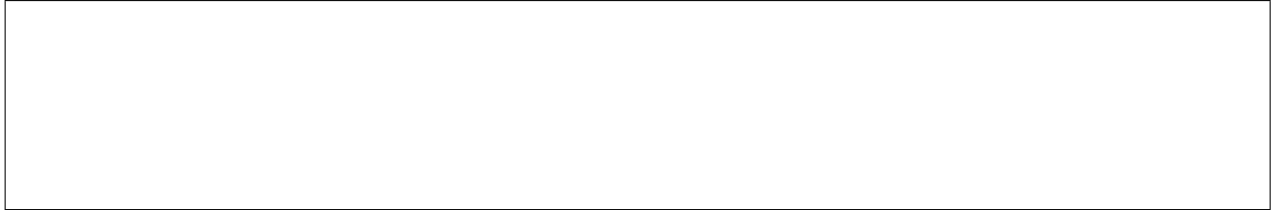
r	b	t	x	m	k
e	a	t	x	d	s
x	b	a	t	o	p
q	y	v	x	d	l

In the examples shown above, ***spl*** and ***vx*** are considered words on the board, even though they are not actual English words. For the purpose of this program, a word in the word search board is any sequence of 1 or more character(s) that appears in a row (from left to right) or column (from top to bottom) of the word search board, regardless of whether it is an actual English word. We will not consider words that appear on a diagonal.

The next two pages contain starter code for a program that continually prompts the user to guess a word, until they guess a word that appears on the word search board.

The functions and main on the next two pages are in the same module. Complete the missing code according to the docstring descriptions and inline comments. Avoid duplicate code by calling functions from this module as helpers. You should do this even if the helpers are incomplete/incorrect. Each function will be marked as though any helper it relies on works correctly.

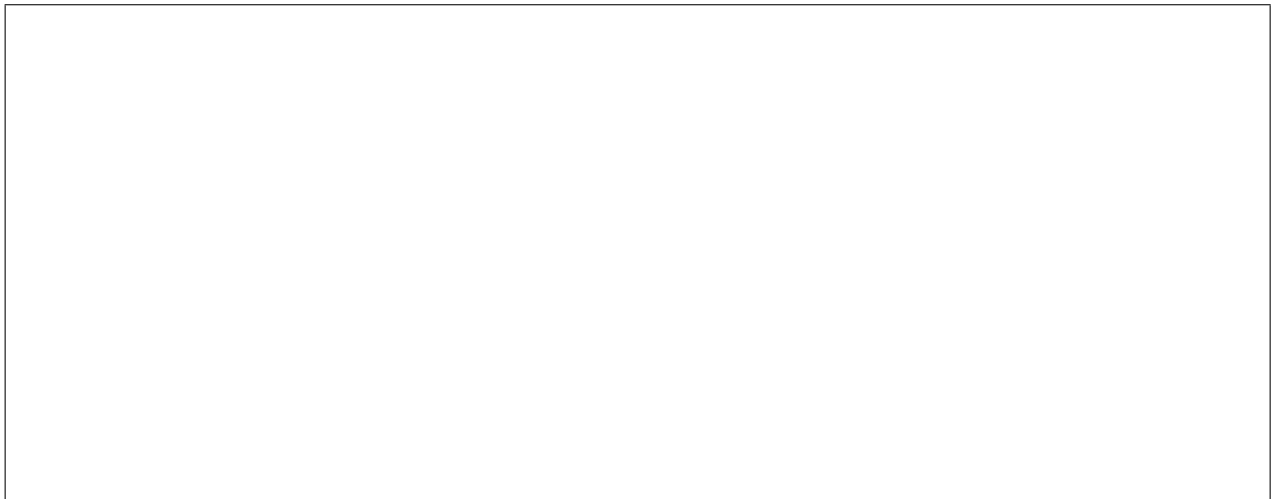
```
def load_board(ws_file):  
    '''(file) -> list of str  
    ws_file is a "word search file". Each line of the file contains  
    lowercase letters and all lines have the same length.  
    Return a "word search list" based on the contents of ws_file.'''
```



```
def display_board(ws_board):  
    '''(list of str) -> NoneType  
    Display the contents of the "word search board" ws_board.'''  
  
    for row in ws_board:  
        print row
```

```
def in_list(L, word):  
    '''(list of str, str) -> bool  
    Return True iff word appears in (is a substring of) at least one item in L.'''  
  
    for item in L:  
        if word in item:  
            return True  
    return False
```

```
def in_row(ws_board, word):  
    '''(list of str, str) -> bool  
    Return True iff word appears in at least one row of "word search board" ws_board.'''
```



```
def in_column(ws_board, word):  
    '''(list of str, str) -> bool  
    Return True iff word appears in at least one column of "word search board" ws_board.'''
```

```
def is_found(ws_board, word):  
    '''(list of str, str) -> bool  
    Return True iff word appears in at least one row or column of "word search board"  
    ws_board.'''
```

```
if __name__ == "__main__":
```

```
    ws_board = load_board("words.txt")  
    display_board(ws_board)
```

```
    # Continually prompt the user to enter a word, until the user enters a  
    # word that appears in the "word search board" ws_board.
```

```
    print "You found a word! The word %s appears in the word search board." % (word)
```

Question 9. [12 MARKS]

This question involves working with dictionaries of the following types:

“book to people dictionary”

- **key:** a book (a `str`), **value:** a list of people who have read that book (a list of `strs`)
- **example:**
`{‘Treasure Island’: [‘Joanna’, ‘Jonathan’, ‘Hui’, ‘Ali’],\n ‘Wuthering Heights’: [‘Hui’, ‘Monia’, ‘Ali’],\n ‘Middlemarch’: [‘Vlad’, ‘Ali’]}`

“person to books dictionary”

- **key:** a person (a `str`), **value:** a list of books that person has read (a list of `strs`)
- **example:**
`{‘Vlad’: [‘Middlemarch’], ‘Monia’: [‘Wuthering Heights’],\n ‘Ali’: [‘Treasure Island’, ‘Wuthering Heights’, ‘Middlemarch’],\n ‘Hui’: [‘Treasure Island’, ‘Wuthering Heights’],\n ‘Joanna’: [‘Treasure Island’], ‘Jonathan’: [‘Treasure Island’]}`

Complete the following functions according to their docstring descriptions.

Part (a) [5 MARKS]

```
def invert_book_to_people(book_to_people):  
    '''(dict of {str: list of strs}) -> dict of {str: list of strs}  
    book_to_people is a "book to people dictionary".  
    Return a "person to books dictionary" based on book_to_people.'''
```

Part (b) [7 MARKS]

This part of the question involves another type of dictionary that is used to make recommendations.

“recommendations dictionary” for book B

- **key:** a book (a `str`) that people who read book B have also read,
value: the number of people (an `int`) who read both this book (the key) and book B
- **example:** for ‘Wuthering Heights’

```
{‘Treasure Island’: 2, ‘Middlemarch’: 1}
```

```
def make_recommendations(book_to_people, book):  
    '''(dict of {str: list of strs}) -> dict of {str: int}  
    book_to_people is a "book to people dictionary" and book is a book title  
    that occurs as a key in book_to_people.  
    Return a new "recommendations dictionary" for book.'''
```

Question 10. [8 MARKS]

Throughout this question, assume that we are sorting lists into non-descending order. **Do not guess.** There is a one-mark deduction for incorrect answers.

Part (a) [2 MARKS]

We are partly through sorting a list, and have completed 3 passes through the data. The list currently contains [4, 6, 7, 8, 5, 9, 7, 15]. Which sorting technique are we definitely *not* using? Circle one.

bubble sort

insertion sort

Part (b) [2 MARKS]

Suppose you have a list of 100 distinct numbers in order from largest to smallest, in other words, backwards to the order we want. Which sorting technique would require the most comparisons? Circle one.

insertion sort

selection sort

they would require an equal number

Part (c) [2 MARKS]

Suppose you have a list of 100 distinct numbers in order from largest to smallest, in other words, backwards to the order we want. Which sorting technique would require the most swaps? Circle one.

insertion sort

selection sort

they would require an equal number

Part (d) [2 MARKS]

Consider the following lists: L1 = [9, 8, 7, 6, 5, 4, 3, 2, 1] and L2 = [8, 1, 3, 6, 9, 7, 4, 2, 5] Which list would cause insertion sort to do more swaps? Circle one.

L1

L2

they would require an equal number

Question 11. [4 MARKS]

Do not guess. There is a half-mark deduction for incorrect answers.

Part (a) [1 MARK]

```
for i in range(len(L)):
    L[i] = L[i] ** 2
```

Let n be the size of the list L . Which of the following most accurately describes how the runtime of this function grow as n grows? Circle one.

- (a) It grows linearly, like n does. (b) It grows quadratically, like n^2 does.
(c) It grows less than linearly. (d) It grows more than quadratically.

Part (b) [1 MARK]

```
i = 0
while i != len(L):
    L[i] = L[i] ** 2
    i += 1
```

Let n be the size of the list L . Which of the following most accurately describes how the runtime of this function grow as n grows? Circle one.

- (a) It grows linearly, like n does. (b) It grows quadratically, like n^2 does.
(c) It grows less than linearly. (d) It grows more than quadratically.

Part (c) [1 MARK]

```
for x in range(len(L)):
    for y in range(len(L[0])):
        if y == y1:
            L[x][y1] = L[x][y1] ** 2
```

Let n be the size of the list L . Which of the following most accurately describes how the runtime of this function grow as n grows? Circle one.

- (a) It grows linearly, like n does. (b) It grows quadratically, like n^2 does.
(c) It grows less than linearly. (d) It grows more than quadratically.

Part (d) [1 MARK]

```
for x in range(len(L)):
    L[x][y1] = L[x][y1] ** 2
```

Let n be the size of the list L . Which of the following most accurately describes how the runtime of this function grow as n grows? Circle one.

- (a) It grows linearly, like n does. (b) It grows quadratically, like n^2 does.
(c) It grows less than linearly. (d) It grows more than quadratically.

Total Marks = 76

*[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:**__builtins__:**`len(x) -> int`

Return the length of the list, tuple, dict, or string x.

`max(L) -> object`

Return the largest value in L.

`min(L) -> object`

Return the smallest value in L.

`open(name[, mode]) -> file`

Open a file. Legal modes are "r" (read), "w" (write), and "a" (append).

`range([start], stop, [step]) -> list of ints`

Return a list containing the integers starting with start and ending with stop - 1 with step specifying the amount to increment (or decrement).

If start is not specified, the list starts at 0. If step is not specified, the values are incremented by 1.

`raw_input([prompt]) -> str`

Read a string from standard input. The trailing newline is stripped.

dict:`D[k] --> object`

Return the value associated with the key k in D.

`k in d --> bool`

Return True if k is a key in D and False otherwise.

`D.get(k) -> object`

Return D[k] if k in D, otherwise return None.

`D.keys() -> list of objects`

Return the keys of D.

`D.values() -> list of objects`

Return the values associated with the keys of D.

`D.items() -> list of (key, value) pairs`

Return the (key, value) pairs of D, as 2-tuples.

file (also called a "reader"):`F.close() --> NoneType`

Close the file.

`F.read([size]) -> read at most size bytes, returned as a str`

If the size argument is negative or omitted, read until EOF (End of File) is reached.

`F.readline([size]) -> next line from the file, as a str. Retain newline.`

A non-negative size argument limits the maximum number of bytes to return (an incomplete line may be returned then). Return an empty string at EOF.

float:`float(x) -> float`

Convert a string or number to a floating point number, if possible.

int:`int(x) -> int`

Convert a string or number to an integer, if possible. A floating point argument will be truncated towards zero.

list:

x in L --> bool
Return True if x is in L and False otherwise.
L.append(x) --> NoneType
Append x to the end of the list L.
L.index(value) -> int
Return the lowest index of value in L.
L.insert(index, x) --> NoneType
Insert x at position index.
L.pop() --> object
Remove and return the last item from L.
L.remove(value) --> NoneType
Remove the first occurrence of value from L.
L.reverse() --> NoneType
Reverse the elements of L.
L.sort() --> NoneType
Sort the list L in ascending order.

str:

x in s --> bool
Return True if x is in s and False otherwise.
str(x) -> str
Convert an object into its string representation, if possible.
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.index(sub) -> int
Like find but raises an exception if sub does not occur in S.
S.isdigit() -> bool
Return True if all characters in S are digits and False otherwise.
S.lower() -> str
Return a copy of the string S converted to lowercase.
S.lstrip([chars]) -> str
Return a copy of the string S with leading whitespace removed. If chars is given and not None, remove characters in chars instead.
S.replace(old, new) -> str
Return a copy of string S with all occurrences of the string old replaced with the string new.
S.rstrip([chars]) -> str
Return a copy of the string S with trailing whitespace removed. If chars is given and not None, remove characters in chars instead.
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.strip() -> str
Return a copy of S with leading and trailing whitespace removed.
S.upper() -> str
Return a copy of the string S converted to uppercase.

media:

copy(Picture) --> Picture

Return a copy of the Picture.

create_picture(int, int) --> Picture

Given a width and a height, return a Picture with that width and height. All pixels are white.

get_blue(Pixel) --> int

Return the blue value of the given Pixel.

get_color(Pixel) --> Color

Return the Color object with the given Pixel's RGB values.

get_green(Pixel) --> int

Return the green value of the given Pixel.

get_pixel(Picture, int, int) --> Pixel

Given x and y coordinates, return the Pixel at (x, y) in the given Picture.

get_red(Pixel) --> int

Return the red value of the given Pixel.

get_x(Pixel) --> int

Return the x coordinate of the given Pixel.

get_y(Pixel) --> int

Return the y coordinate of the given Pixel.

set_blue(Pixel, int) --> NoneType

Set the blue value of the given Pixel to the given int value.

set_color(Pixel, Color) --> NoneType

Set the RGB values of the given Pixel to those of the given Color.

set_green(Pixel, int) --> NoneType

Set the green value of the given Pixel to the given int value.

set_red(Pixel, int) --> NoneType

Set the red value of the given Pixel to the given int value.