Question 1. [8 marks]

Each subquestion below has a small piece of code that is supposed to print either 'BINGO' or 'B I N G O', but has a small part missing. For each one, add the missing part inside the box. Your solution must follow the instructions in the comment statement. Each subquestion is independent.

Part (a) [1 mark]

```python
letters = ['B', 'I', 'N', 'G', 'O']
# Iterate over the list to print B I N G O on one line
for char in letters: print char, print
```

Part (b) [1 mark]

```python
games = ['Monopoly', 'Tic Tac Toe', 'Bridge']
# Replace 'Bridge' with 'BINGO'
games[-1] = 'BINGO'
print games.pop()
```

Part (c) [1 mark]

```python
games = [['Bridge', 'Poker'], ['BINGO', 'Euchre'], ['Monopoly', 'Clue']]
# Print the appropriate element of this data structure
print games[1][0]
```

Part (d) [1 mark]

```python
pets = {'cat': 'Whiskers', 'fish': 'Nemo', 'dog': 'BINGO'}
# Print the correct dictionary value
print pets['dog']
```
Part (e) [1 mark]
more_letters = 'XXBIXNGXOX'
# Iterate over the string, but don’t print any of the X's

for char in more_letters:
    if char != 'X':
        print char,

Part (f) [1 mark]
# some_string is a string variable with a value that you don’t know
some_string = raw_input('enter something')
s = 'axBINxx' + some_string + 'xGOmn'
# Extract the slices that you need from s and build slice
slice = s[2:5] + s[-4:-2]
print slice

Part (g) [1 mark]
letters = ['A', 'B', 'G', 'I', 'M', 'Nope', 'O']
# Using elements (or pieces of elements) from letters, build string result
print result

Part (h) [1 mark]
x = -2 + 4 / 2 + 1
# Make a list with only one element that is 'BINGO' and other elements with other values
# so that the print statement prints the one 'BINGO' element
my_list = [x, "BINGO"]
print my_list[x]
Question 2. [4 marks]
In the box beside each piece of code below, write its output. If it would generate an error, say so, and give
the reason for the error.

Part (a) [1 mark]

```python
L = [3, 2, 1, 0]
for item in L:
    item = item + 1
print L
```

Part (b) [1 mark]

```python
L = [10, 11, 12, 13]
for item in L:
    L[item] = L[item] + 1
print L
```

Part (c) [1 mark]

```python
L = [5, 6, 7, 8]
for item in range(len(L)):
    L[item] = L[item] + 1
print L
```
Part (d)  [1 MARK]

```python
L = [2, 4, 6, 8]
for item in range(len(L)):
    item = item + 1
print L
```

Solution and Marking Scheme:

A. [3, 2, 1, 0]
B. Error: there is no index [10] in the list OR item > len(L)
   1/2 for "This would cause an error because you cannot iterate
      over a list using its elements as index values"
   1/2 for item can’t be used as the key
C. [6, 7, 8, 9]
D. [2, 4, 6, 8]

overall deduction of 1 for omitting the [] on lists in multiple parts of question
Question 3. [8 marks]
In the box beside each piece of code below, write its output. If it would generate an error, say so, and give the reason for the error.

Part (a) [2 marks]

```
L1 = ["I", "like", "candy"]
L2 = L1.append("somewhat")
print L1
print L2
```

Part (b) [2 marks]

```
n1 = 96
n2 = n1
n2 = n1 + 1
print n1
print n2
```

Part (c) [2 marks]

```
x = [1, 2]
L1 = [x, [8, 9]]
L2 = L1[:]
L2[0][1] = 999
print L1
print L2
```
Part (d) [2 marks]

def f(s):
    s = s * 2
    return s

food = "pizza"
f(food)
print food
print f('cheese')

Solution and Marking Scheme:

A. ['I', 'like', 'candy', 'somewhat']
   None

B. 96
   97

C. [[1, 999], [8, 9]]
   [[1, 999], [8, 9]]

D. pizza
   cheese cheese

   pizzapizza
   cheese cheese gets 1/2

   pizzapizza
   pizza
   cheese cheese gets 1.5/2
Question 4. [7 marks]

This question involves a dictionary of the same format as the actor to movie dictionary that you used in A3.

Part (a) [4 marks]

Complete the function below according to its docstring.

```python
def largest_cast(actor_dict, movie1, movie2):
    '''Return which of two movies had the largest number of actors. Each key in dictionary actor_dict is the name of an actor (a string) and its value is a list of the movies in which that actor has performed. (Each movie name is a string also.) Return movie1 or movie2, whichever has the larger number of actors. If it is a tie, return either. '''

    movie1_count = 0
    movie2_count = 0
    for actor in actor_dict:
        for movie in actor_dict[actor]:
            if movie == movie1:
                movie1_count += 1
            elif movie == movie2:
                movie2_count += 1

    # alternate solution
    # if movie1 in actor_dict[actor]:
    #     movie1_count += 1
    # if movie2 in actor_dict[actor]:
    #     movie2_count += 1

    if movie1_count > movie2_count:
        return movie1
    else:
        return movie2
```

Part (b) [1 mark]

Assume that the number of actors in the dictionary is $n$, and that each actor on average performs in $m$ movies. How many string comparisons (checking if one string equals another) does your solution perform?  

**Important Note:** If you used the boolean expression `x in L` for some $x$ on some list $L$, you should count this as performing $\text{len}(L)$ comparisons.

$n \times m$

Part (c) [1 mark]

Suppose that $n$ and $m$ are large and that you needed to do this operation (comparing the cast sizes of two movies) many many times. You could design your program to save time. Explain in one or two sentences how you would do this.

Create and keep a dictionary that maps movies to actors (i.e. invert the actor dictionary)
Part (d)  [1 mark]
Explain briefly why this efficiency improvement only makes sense if you are comparing cast sizes more than once.

Inverting the dictionary takes $m \times n$ operations. So, if you compare cast sizes just once, you might as well just do the $m \times n$ operations required when using the actor dictionary directly. When you compare cast sizes multiple times, the $m \times n$ cost of the inversion only has to be done once, after which you can find the two cast sizes in time linear in $n$. 
Question 5. [8 marks]
The parts of this question are completely independent of each other. You can solve one without solving the other.

Part (a) [3 marks]
Consider the following function:

```python
def min_value(L):
    '''L is a list of ints that are >= -1. Return the minimum value in L that is > -1. If L doesn't have any value in it other than -1, return -1.'''
```

Suppose that we want to test `min_value`. Describe three test cases that each test different “categories” of inputs. To describe each test case, give the list that you would pass to `min_value`, 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 three more.

<table>
<thead>
<tr>
<th>Value of L</th>
<th>Return value</th>
<th>Purpose of this test case</th>
</tr>
</thead>
<tbody>
<tr>
<td>[]</td>
<td>-1</td>
<td>empty list</td>
</tr>
</tbody>
</table>

Solution:

Lots of options, including:
list of length 1
list longer than length 1
all elements equal -1
one element is -1
position of min at beginning, middle, end
Part (b)  [5 MARKS]

Now write the function according to its docstring specification.

def min_value(L):
    '''L is a list of ints that are >= -1. Return the minimum value in L that is > -1. If all the values in L are -1, return -1.'''

Solution:

def min_value (L):
    '''L is a list of ints that are >= -1. Return the minimum value in L that is > -1. If L doesn't have any value in it other than -1, return -1.'''

    ans = -1
    for n in L:
        if n > -1:
            if ans == -1:
                ans = n
            else:
                ans = min(ans, n)

    return ans


**Question 6.** [4 marks]

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

The Toronto Maple Leafs hockey team plays games on many days but not every day. The program below asks the user to say what happened on every single day, whether a game was played or not. The user either answers “a win”, “a loss” or “no game”. The program keeps asking for the next day’s results until it gets two consecutive days where the Leafs played and had the same result (either two wins on back-to-back days or two losses on back-to-back days).

Fill in the missing while loop condition to make the program work as described above.

```python
previous = raw_input("First day’s game result? ")
current = raw_input("Next day’s game result? ")

while previous == current and current != "no game":
    previous = current
    current = raw_input("Next day’s game result? ")
print 'Finally! Two days in a row with the same game result.'
```

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

Consider the following for loop:

```python
for i in range(0,77):
    print i
```

Re-write it to do the same thing using a while loop.

**Solution:**

```python
i = 0
while i < 77:
    print i
    i += 1
```
Question 7. [6 MARKS]
Consider the following function.

```python
def average_word_length (text):
    '''text is a str consisting only of words and spaces (no punctuation). text
    is guaranteed to have at least one word. There can be one or more than one
    whitespace character between pairs of words. Return the average length of
    all words in text, as a float.'''
```

Part (a) [1 MARK]
What should the function return if given this string: "\t I am the\n\n walrus"?

Solution:

3.0

Part (b) [5 MARKS]
Write function `average_word_length`. You don’t need to repeat the docstring. Part of your mark will be for avoiding unnecessarily complicated techniques.

```python
def average_word_length (text):
    '''text is a str consisting only of words and spaces (no punctuation). text
    is guaranteed to have at least one word. There can be one or more than one
    whitespace character between pairs of words. Return the average length of
    all words in text as a float.'''

    count = 0
    sum = 0
    for token in text.split():
        count += 1
        sum += len(token)
    return sum / float(count)
```

```python
print average_word_length("\t I am the \n\n walrus")
```
Question 8. [9 marks]

Part (a) [6 marks]
Consider the following function `puzzle` where we don’t know the type of parameter `thingy`.

```python
def puzzle(thingy):
    for i in range(1, len(thingy)):
        if thingy[i-1] < thingy[i]:
            return i-1
    return -1
```

• Will the function work if `thingy` is type `string`? Pick one.
  YES [ ]  NO [ ]
  If you answered yes, give a test case showing both a value for string `s` where `len(s) >= 3`, and the result of calling `puzzle(s)`. If you answered no, explain why calling `puzzle` with a string argument would give an error.

• Will the function work if `thingy` is type `int`? Pick one.
  YES [ ]  NO [ ]
  If you answered yes, give a test case showing both a value for integer `x` where `x >= 3`, and the result of calling `puzzle(x)`. If you answered no, explain why calling `puzzle` with an integer argument would give an error.

• Will the function work if `thingy` is type `list`? Pick one.
  YES [ ]  NO [ ]
  If you answered yes, give a test case showing both a value for list `L` where `len(L) >= 3`, and the result of calling `puzzle(L)`. If you answered no, explain why calling `puzzle` with a list argument would give an error.
- Function **puzzle** will work correctly if called with a dictionary argument, but only for a carefully constructed dictionary. Give an example of a dictionary \( d \) where \( \text{len}(d) \geq 3 \), where calling **puzzle**(\( d \)) would not result in an error.

**Solution:**

A. Yes; if called with string `'abc'`, the result is 0.
B. No; ints cannot be indexed.
C. Yes; if called with list `[a', 'b', 'c']`, the result is 0.
D. `{1: 'a', 2: 'b', 3: 'c'}`

**Part (b) [3 MARKS]**

Write a good docstring for this function.

def enigma(s, c, n):
    '''Return a copy of string s, with each of the first n occurrences of character c replaced by 'X'.'''
Question 9. [8 marks]

Consider the following function:

```python
def mystery(d1, d2):
    ans = {}
    for key, value in d1.items():
        if key in d2:
            ans[key] = max(value, d2[key])
        else:
            ans[key] = value
    for key, value in d2.items():
        if key in d1:
            ans[key] = max(value, d1[key])
        else:
            ans[key] = value
    return ans
```

Part (a) [1 mark]

What value is returned by this call to the function:

```python
mystery({1:2, 4:5, 3:44, 10:11}, {4:55, 8:9, 3:4, 1:2})
```

The order of the key-value pairs in your answer does not matter.

Solution:

```
{8: 9, 1: 2, 10: 11, 3: 44, 4: 55}
```

Part (b) [3 marks]

Write a good docstring for this function.

Solution:

Return a new dictionary that is the result of merging dictionaries d1 and d2. For every key that is in both dictionaries, give it the larger of the two values associated with that key in d1 and d2. Include key-value pairs that occur in only one of the two dictionaries.
Part (c)  [4 MARKS]
The two loops in this function are almost the same. With the right helper function, mystery could be written as shown at the bottom of this page, avoiding repetitive code. In the space provided, write a helper function that makes this possible. Then complete mystery by adding two appropriate calls to your helper. Your new version of mystery must do the same thing as our original version.

You do not need to write a docstring for your helper function.

def mystery(d1, d2):
    ans = {}
    # Call your helper function once to replace the first loop from the original function:
    # Call your helper function once to replace the first loop from the original function:
    # Call your helper function once to replace the first loop from the original function:

    return ans
Solution:

def update_with_larger(new, d1, d2):
    for key, value in d1.items():
        if key in d2:
            new[key] = max(value, d2[key])
        else:
            new[key] = value

def mystery(d1, d2):
    ans = {}
    # Call your helper function once to replace the first loop from the original function:
    helper(ans, d1, d2)
    # Call your helper function once to replace the first loop from the original function:
    helper(ans, d2, d1)
    return ans
Question 10. [7 MARKS]
Complete the following function according to its docstring specification. For this question, a blank line is defined to be a line with no characters on it — not even spaces or tabs.

You must not read the whole file at once into a single string. You must read one line at a time, or your mark will be zero.

```python
def longest_sequence(r):
    '''Return the number of lines in the longest consecutive sequence of
    blank lines in open reader r, or zero if there are no blank lines at all.'''

    # The longest sequence we've seen has no length at all -- we haven't seen
    # one!
    max_sequence = 0

    line = r.readline()
    on_sequence = False
    while line:
        if line == '\n':
            if on_sequence:
                # It's a blank line and it's continuing a sequence
                sequence_length += 1
            else:
                # It's the first blank line in a sequence
                sequence_length = 1
                on_sequence = True
        else:
            if on_sequence:
                # It's the first non-blank line after a sequence
                if sequence_length > max_sequence:
                    max_sequence = sequence_length
                on_sequence = False
                # If it's a non-blank line that's not at the end of a sequence,
                # do naught!
                line = r.readline()
        return max_sequence
```

Bad Solution: doesn't work when the file ends with the longest sequence

```python
def longest_sequence(r):
    '''Return the number of lines in the longest sequence of blank lines in
    open reader r, or zero if there are no blank lines at all.'''

    # The longest sequence we've seen has no length at all -- we haven't seen
    # one!
    max_sequence = 0

    line = r.readline()
    on_sequence = False
    while line:
        if line == '\n':
            if on_sequence:
                # It's a blank line and it's continuing a sequence
                sequence_length += 1
            else:
                # It's the first blank line in a sequence
                sequence_length = 1
                on_sequence = True
        else:
            if on_sequence:
                # It's the first non-blank line after a sequence
                if sequence_length > max_sequence:
                    max_sequence = sequence_length
                on_sequence = False
                # If it's a non-blank line that's not at the end of a sequence,
                # do naught!
                line = r.readline()
    return max_sequence
```

Good Solution: does work when the file ends with the longest sequence

```python
def longest_sequence(r):
    '''Return the number of lines in the longest sequence of blank lines in
```
open reader r, or zero if there are no blank lines at all.'''

longest = 0
current = 0
for line in r:
    if line == '\n':
        current += 1
    else:
        # we have possibly just ended a sequence, so check if longest
        if current > longest:
            longest = current
        current = 0
# in case the longest sequence is the last one, we need to check once more
if current > longest:
    longest = current
return longest
Question 11.  [6 MARKS]
Throughout this question, assume that we are sorting a list into non-descending order. Each subquestion below shows the contents of the list before sorting begins, and after each phase of the sorting process. Identify which sorting technique is being used. Do not guess. There is a one-mark deduction for incorrect answers.

Part (a)  [2 MARKS]
[5, 9, 0, 4, 6, 8, 2]  
[5, 0, 4, 6, 8, 2, 9]  
[0, 4, 5, 6, 2, 8, 9]  
[0, 4, 5, 2, 6, 8, 9]  
[0, 4, 2, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
Which sort technique are we using? Check one.

☐ selection sort  ☐ insertion sort  ☐ bubble sort

Part (b)  [2 MARKS]
[5, 9, 0, 4, 6, 8, 2]  
[5, 9, 0, 4, 6, 8, 2]  
[5, 9, 0, 4, 6, 8, 2]  
[0, 5, 9, 4, 6, 8, 2]  
[0, 4, 5, 9, 6, 8, 2]  
[0, 4, 5, 6, 9, 8, 2]  
[0, 4, 5, 6, 8, 9, 2]  
[0, 2, 4, 5, 6, 8, 9]  
Which sort technique are we using? Check one.

☐ selection sort  ☐ insertion sort  ☐ bubble sort

Part (c)  [2 MARKS]
[5, 9, 0, 4, 6, 8, 2]  
[0, 9, 5, 4, 6, 8, 2]  
[0, 2, 5, 4, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
[0, 2, 4, 5, 6, 8, 9]  
Which sort technique are we using? Check one.

☐ selection sort  ☐ insertion sort  ☐ bubble sort

Solution:  A. Bubble B. Insertion C. Selection
Short Python function/method descriptions:

__builtins__:
   len(x) -> integer
       Return the length of the list, tuple, dict, or string x.
   max(L) -> value
       Return the largest value in L.
   min(L) -> value
       Return the smallest value in L.
   open(name[, mode]) -> file object
       Open a file. Legal modes are "r" (read), "w" (write), and "a" (append).
   range([start], stop, [step]) -> list of integers
       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]) -> string
       Read a string from standard input. The trailing newline is stripped.

dict:
   D[k] --> value
       Return the value associated with the key k in D.
   k in d --> boolean
       Return True if k is a key in D and False otherwise.
   D.get(k) -> value
       Return D[k] if k in D, otherwise return None.
   D.keys() -> list of keys
       Return the keys of D.
   D.values() -> list of values
       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()
       Close the file.
   F.read([size]) -> read at most size bytes, returned as a string.
       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 string. 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) -> floating point number
       Convert a string or number to a floating point number, if possible.

int:
   int(x) -> integer
       Convert a string or number to an integer, if possible. A floating point
       argument will be truncated towards zero.

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

Module random:
    randint(a, b)
        Return random integer in range [a, b], including both end points.

str:
    x in s --> boolean
        Return True if x is in s and False otherwise.
str(x) -> string
    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]) -> integer
    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) -> integer
    Like find but raises an exception if sub does not occur in S.
S.isdigit() -> boolean
    Return True if all characters in S are digits and False otherwise.
S.lower() -> string
    Return a copy of the string S converted to lowercase.
S.lstrip([chars]) -> string
    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) -> string
    Return a copy of string S with all occurrences of the string old replaced with the string new.
S.rstrip([chars]) -> string
    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 strings
    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() -> string
    Return a copy of S with leading and trailing whitespace removed.
S.upper() -> string
    Return a copy of the string S converted to uppercase.