Family Name:..
Given Names: ....
Student ID: .... Username: ....

University of Canterbury

End of Year Examinations 2011

Combined Question and Answer Book

Prescription Number: COSC121S2
Paper Title: Introduction to Computer Programming

Time Allowed: THREE (3) HOURS
Number of Pages: 14 (plus 1 extra page for answers that overflow)

For use by markers only

Question	Out Of	Mark
1	17	
2	17	
3	12	
4	18	
5	14	
6	16	
7	6	
TOTAL	100	

Note:

- Attempt ALL questions.
- This is a CLOSED BOOK exam.
- Calculators are not permitted.
- Indicated marks are out of a total of 100 marks.
- This is a combined question and answer book. Answer each question in the space provided within the question.
- Extra space is available at the end for answers that overflow the provided answer boxes.
- Assume the use of Python 2.6 (the version used in the course) throughout the exam.
- Unless otherwise specified, you may make use of any standard Python library functions.
- When using library functions, you do not have to explicitly *import* the module(s).
- Some of the following methods, constructors and variables may be useful. Parameters in square brackets are optional and [, ...] denotes a sequence of optional extra parameters.

Module/class	***Function/Method***	***Brief description***
`__builtins__`	`len(seq)`	Length of sequence 'seq'
`__builtins__`	`raw_input([prompt])`	Prints *prompt* if given then reads and returns a string entered via the keyboard
`__builtins__`	`sum(seq [,...])`	The sum of the elements of sequence *seq* if it's the only parameter else the sum of all the parameters.
`__builtins__`	`min(seq [,...])`	The minimum of the elements of sequence *seq* if it's the only parameter else the minimum of all the parameters.
`__builtins__`	`max(seq [,...])`	The maximum of the elements of sequence *seq* if it's the only parameter else the maximum of all the parameters.
`__builtins__`	`sorted(seq [,...])`	Returns a new list that is the sorted sequence *seq*.
`__builtins__`	`reversed(seq)`	Returns a reverse order iterator.
`str`	`S.find(sub)`	The index to the first occurrence of *sub* in S or -1 if *sub* is not found
`str`	`S.lower()`	A lower-case copy of S
`str`	`S.upper()`	An upper-case copy of S
`str`	`S.strip([chars])`	A copy of S with leading and trailing whitespace removed (or leading and trailing characters in *chars,* if given)
`str`	`S.split([sep])`	A list of substrings of S separated by *sep* if given or by whitespace if not.
`str`	`S.join(seq)`	Joins items in *seq* with the string S and returns the result.
`list`	`L.append(item)`	Appends *item* to L, returns None
`List`	`L.pop()`	Remove the item from the end of L and return it
`list`	`L.count(item)`	A count of the occurrences of *item* in L

Question 1. [17 marks]

Fill in the blank spaces in the following statements. Most (but not all) spaces require just a single character, number or word. If the answer is a Python string, write it enclosed in single or double quote characters, e.g. as `'My answer'`.

Each blank space is worth **1 mark**.

a) `24 - 3 * 3` evaluates to ____________________.

b) `12 / 3.0` evaluates to ____________________.

c) `("6" + "2") * 3` evaluates to ____________________.

d) `13 / 6` evaluates to ____________________.

e) `13 % 6` evaluates to ____________________.

f) After the assignment `x = range(5,10)`, `x[0]` has the value ______________ , `x[3]` has the value ____________, `x[-1]` has the value ____________, `x[1:3]` has the value ________________, `x[-2:]` has the value _____________, `x[-3:-1]` has the value ______________ and `x[:]` has the value _________________.

g) `tuple(list("abc"))` evaluates to ____________________.

h) `"(x = %.3fy)" % 3.14159` evaluates to ____________________.

i) `'7' < '8' < '6'` evaluates to ____________________.

j) `7 < 8 and not 8 < 7 or 8 < 7` evaluates to ____________________.

k) `not(not a and not b)` can be most simply written as _________________________.

Question 2. [17 marks]

Write the output from each of the following programs in the underlined space provided.

IMPORTANT: If there are multiple lines of output, you should still write your answer in the single-line space provided, using '\n' to indicate a line break. You don't need to include the final '\n' at the end of the output. For example, the output from the program.

```
print "Hi"
print "Wendy"
```

should be written in the underlined space as `Hi\nWendy`

a) **[3 marks]**

```
t = 5
u = "hi"
t = 3 + t
v = t + len(u)
print t, u, v
```

Output: ______________________________

b) **[3 marks]**

```
s1 = 3 * "hi"
s2 = "lo"
s2 = s1 + 3 * s2
s3 = s2[2:-2]
print "s3 = '%s' (length %i)" % (s3, len(s3))
```

Output: ______________________________

c) **[5 marks]**

```
vals = [5, 4, 6, 7, 2]
s = 0
c = 0
for v in vals:
    if v % 2 == 1:
        s += v
        print v,
    else:
        c = c + 1
        print '-',
print s,c
```

Output: ______________________________

d) **[6 marks]**

```
def p(s):
    t = []
    r = []
    for c in s:
        if c in '+-':
            while len(t) > 0:
                r.append(t.pop())
            t.append(c)
        elif c in '*/':
            while len(t) > 0 and t[-1] in '*/':
                r.append(t.pop())
            t.append(c)
        else:
            r.append(c)
    while len(t) > 0:
        r.append(t.pop())
    return ''.join(r)

print p('6+5*4-3/2')
```

Output: ____

Question 3. [12 marks]

The question concerns output produced following calls to functions `foo` and `boo` shown below. In your output, show spaces as ␣, e.g. "`Angus␣McGurkinshaw`". All programs run without errors.

```
def foo (val):
    val *= 3
    return val

def boo(val):
    val = val * 3
    return val
```

a) **[3 marks]**

```
aval = "A"
bval = foo(aval)
print aval, bval
```

Output: ____

b) **[3 marks]**

```
aval = [1]
bval = foo(aval)
print aval, bval
```

Output: ____

c) **[3 marks]**

```
aval = "A"
bval = boo(aval)
print aval, bval
```

Output: ____

d) **[3 marks]**

```
aval = [1]
bval = boo(aval)
print aval, bval
```

Output: ____

Question 4. [18 marks]

A Python dictionary is used to keep track of the number of occurrences of car license plate numbers in a file. Each line of the file contains a single string recording a license plate number.

a) **[6 marks]** Complete the following function by filling in the blanks.

```
def make_cruising_dict(file_name):

    '''Opens file_name for reading. Each line of the file contains
    a single string that is a car license plate number. Returns a
    dictionary containing key/value pairs showing, for each license
    plate, the number of times it occurred in the file.'''

    plate_count = ______________________# make an empty dictionary

    infile = open(file_name)

    for line in _________________________:

        plate = line.strip()

        plate_count________ = __________________________________

    infile.close()

    return _________________________
```

b) **[8 marks]** The following function returns an inverted dictionary, so each key is the number of times vehicles have been seen, and each value is a list of vehicles seen that number of times. Compete the function.

```
def invert_dict(plate_count):
    '''Return inverted dictionary. For example, if plate_count is
    {'AMR576':5, 'CMN555':5, 'BGK444':2}
    then the key/value pairs of the returned dictionary
    might be (ordering of the key/value pairs is unknown)
    {2:['BGK444'], 5:['AMR576', 'CMN555']} '''

    count_plate = _________________________

    for (plate, count) in ____________________________ :

        if count in count_plate:

            _______________________________________

        else:

            __________________________________________

    return _____________________
```

c) **[4 marks]** The following function prints to the standard output the content of the inverted dictionary in increasing order of occurrence and in alphabetical order of plate. For example, the dictionary `{5:['CMN555', 'AMR576'], 2:['BGK444']}` would be printed as follows:

```
2
    BGK444
5
    AMR576
    CMN555
```

```
def print_dict(count_plate):

    for count in ____________________________________________:

        print count

        for plate in ____________________________________________:

            print "    ", plate
```

Question 5. [14 marks]

Doctests specify how the following functions should behave. In each case, write the full function (do not repeat the doctests). Note that in our examples, the parameters are replaced with block capitals 'SOMETHING GOES HERE'. Your solution should show the full parameters.

a) **[5 marks]**

```
def stripper(SOMETHING GOES HERE):
    '''Returns a string that is the first argument with all
    occurrences of characters contained in an optional second
    argument removed. If the second argument is absent, then the
    second argument defaults to a space.
    >>> stripper(' A B C ')
    'ABC'
    >>> stripper(' A B C ', 'Z')
    ' A B C '
    >>> stripper('aA bB cC', ' ABC')
    'abc'
    '''
```

b) **[5 marks]**

```
def reverse_concat(SOMETHING GOES HERE):
    '''Returns the result of concatenating any number of string
       arguments in the reverse order.
    >>> reverse_concat('aA')
    'aA'
    >>> reverse_concat('aA', 'bB', 'cC')
    'cCbBaA'
    >>> reverse_concat('Mary', 'had', 'a', 'little', 'lamb')
    'lamblittleahadMary'
    '''
```

c) [4 marks]

```
def sandwich_order(SOMETHING GOES HERE):
    '''Return a string describing a sandwich order.
    >>> sandwich_order()
    'BaconLettuceTomato'
    >>> sandwich_order(meat='Chicken', cheese='Cheese')
    'ChickenCheeseLettuceTomato'
    >>> sandwich_order(tomato='')
    'BaconLettuce'
    >>> sandwich_order(lettuce='', meat='', cheese='Cheese',
                       gherkin='Gherkin')
    'CheeseTomatoGherkin'
    '''
```

Question 6. [16 marks]

Consider the functions `caller` and `fuddle` below. Recall that dividing by zero raises a ZeroDivisionError (which is not a subclass of ValueError).

```
def fuddle(inval):
    if inval == 42:
        raise ValueError("A 42 error!")
    print "Point A"
    try:
        if inval == 1:
            return 1.0/(inval-1)
        if inval < 10:
            raise ValueError("Too Low")
        print "Point B"
    except ValueError, e:
        print "Point C handled", e
    print "Point D"
    return 1.0/inval

def caller(x):
    try:
        result = fuddle(x)
        print "Point E %.1f" % result
    except:
        print "Point F, handled"
    print "Point G"
```

a) **[4 marks]** Show the output produced when `caller` is invoked with the argument value 1, as follows: `caller(1)`

b) **[4 marks]** Show the output produced when `caller` is invoked with the argument value 42, as follows: `caller(42)`.

c) **[4 marks]** Show the output produced when `caller` is invoked with the argument value 5, as follows: `caller(5)`.

d) **[4 marks]** Show the output produced when `caller` is invoked with the argument value 10, as follows: `caller(10)`.

Question 7. [6 marks]

The class `MyInt`, shown below, forms the basis of a mutable version of the built in `int` type. The value it stores is retrieved through the method `getVal` and set through the constructor or the `setVal` method.

```
class MyInt():
    def __init__(self, v):
        self.val = v

    def __str__(self):
        return str(self.val)

    def getVal(self):
        return self.val

    def setVal(self, v):
        self.val = v
```

a) **[3 marks]** Write a method that will allow MyInt objects to be added using the "+" operator.

b) **[3 marks]** Write a method that will allow MyInt objects to be added using the assignment-addition operator ('+=').

END OF EXAMINATION

Extra space for answers that overflow or for corrections, working etc.

If you want an answer here to be marked, you *must* put the question number in the left margin and write *SEE PAGE 14* beside the blank space where the answer would normally go.