CSC 108H1 Y 2011 Midterm
Duration — 90 minutes
Aids allowed: none

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

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

Lecture Section: L0101 **Instructor: Marek Janicki**

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This midterm consists of 6 questions on 13 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Comments 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.
If you use any space for rough work, indicate clearly what you want marked.

1: _____/10
2: _____/ 5
3: _____/ 7
4: _____/ 8
5: _____/ 7
6: _____/ 8

TOTAL: _____/45

Question 1. [10 MARKS]

The following code runs without errors:

```
def func1(n):
    n = n * 2
    return n

def func2(s):
    s = s + ", really"

def func3(l):
    l[0]=1

def func4(l):
    l = [1,2,3]
    return l

def func5(l):
    l = l.append(4)
    return l

if __name__ == "__main__":

    number = 10
    number = func1(number)
    print number

    s = 'I like trees'
    func2(s)
    print s

    list1 = [0,0,0]
    func3(list1)
    print list1[0]

    list2 = [4,5,6]
    func4(list2)
    print list2

    list3=[7,8,9]
    func5(list3)
    print list3
```

On the following page, show the five lines of output that this code produces. **Strong hint:** Use the blank space provided to trace the code using the memory model.

Output produced:

Question 2. [5 MARKS]

Each of these subquestions contains a block of code. Treat each block of code independently (code in one question is not related to code in another), and fill in the blanks for each question.

Part (a) [1 MARK] Order of Execution

```
var_A = [5]
var_B = var_A
var_A[0] = 10
```

After this code is executed, the value of `var_B` is _____.

Part (b) [2 MARKS] Conditionals and Booleans

The table to the right shows one way in which one might decide when to buy groceries. If you have an empty fridge, then you need to buy groceries, as is the case if you haven't bought groceries for a week in which case you may need to replenish some of the things that go bad. Assume that you have an `int` variable `last_groceries` that tells you how many days it's been since you've bought groceries and a `bool` variable `empty` that is True iff the fridge is empty. Fill in the boolean conditions in the code below to determine what to print to the screen.

	Time from last grocery run	
Fridge empty	< 7 days	$>= 7$ days
Empty	Buy Groceries	Buy Groceries
Not Empty	Wait	Buy Groceries

```
if ____________________________________________________________:
    print 'Buy Groceries'

elif ____________________________________________________________:
    print 'Wait'

else:
    print 'Buy Groceries'
```

Part (c) [1 MARK] Data Types

Fill in the blank so that when this code is run, the user is asked to enter their weigh in kg and height in m. Their BMI is then printed. The BMI is: (their weight in kg)/(their height in m)^2. The user input may contain decimal values (e.g., `1.75`).

```
num1 = raw_input("Please enter your weight in kg: ")
num2 = raw_input("Please enter your height in m: ")

print "Your BMI is " , ________________________________________
```

Part (d) [1 MARK] **Calling Functions**

Fill in the blank to call `average_rainfall` to obtain the average rainfall of Moose Jaw in 2010.

```
def average_rainfall(city, year):
    '''Return the average rainfall of str city in int year.'''
    # The code for this function is not shown.
    return average

print "In 2010, the average rainfall in Moose Jaw was" , ____________________________________
```

Question 3. [7 MARKS]

Write the following function according to its docstring.

```
def evens_and_odds (L):
  '''L is a list. Return a tuple whose first element is
  a list of the elements of L with even indices, and whose second
  element is a list of the elements of L with odd indices. Forexample
  evens_and_odds([1,2,3,4,5,4,3]) should return
  ([1,3,5,3], [2,4,4])'''
```

Question 4. [8 MARKS]

Write the function below, according to its docstring. You must not use a for-loop in this question or your solution will earn zero.

```
def first_float(L):
    '''L is a list of floats.  Return the index of the first element of L that
    has a non-zero component after the decimal. (ie. the first element
    that does not represent an integer). If there is no such element, return -1.
    For example first_float([1.0, 2.0, 1.0, 1.5, 2.0]) will return 3.'''
```

Question 5. [7 MARKS]

Complete the following function according to its docstring description.

```
def ints_only(d):
  '''d is a dict whose values can be of any type.
  Return a new dictionary which contains only the key
  value pairs from d whose values are of type int.
  For example, ints_only({5:[2,3], 4:5, 20: '1'}) would
  return {4:5}'''
```

Question 6. [8 MARKS]

Write the function below, according to its docstring.

```
def goal_scorers(filename):
    '''str filename is the name of a file that stores the number of goals scored by each
    player in a season. Each goal total is stored in a single line as an amount followed by
    the character g (for example: 34g). Return the number of players that scored more than
    20 goals in a season.'''
```

[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) -> integer
    Return the length of the list or string x.
  max(L) -> value
    Return the largest value in L.
  open(name[, mode]) -> file object
    Open a file.
  range([start], stop, [step]) -> list of integers
    Return a list containing the integers starting with stop and ending witt  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.
  type(o) -> o's type
    Return the type of the object o.
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.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 2-tuples.
    Return a list of D's (key, value) pairs.
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 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 then be returned).  Return an empty string at EOF.
float:
  float(x) -> float
    Convert a string or number to a float, if possible.
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.sort(): Sorts the list in ascending order.
int:
  int(x) -> integer
    Convert a string or number to an integer, if possible.  A floating point argument
    will be truncated towards zero.
```

Continued on reverse

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

```
str:
  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 [,start [,end]]) -> int
    Like S.find() but raise ValueError when the substring is not found.
  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.startswith(prefix) -> bool
    Return True if S starts with the specified prefix and False otherwise.
  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.
```

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