UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, December 2014

DURATION: 2½ hours

CSC 180 H1F — Introduction to Computer Programming

Calculator Type: None
Exam Type: D

Aids allowed: reference sheet distributed with the exam
Examiner(s): M. Guerzhoy

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 9 questions on 24 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy is complete, fill in the identification section above, and write your student number where indicated at the bottom of every odd-numbered page (except page 1).*

Answer each question directly on this paper, in the space provided, and use the reverse side of the previous page for rough work. If you need more space for one of your solutions, use the reverse side of a page or the pages at the end of the exam and *indicate clearly the part of your work that should be marked.*

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do—part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

When writing code, assume all input is valid unless otherwise indicated (*i.e.*, do not waste time writing code to error-check). Comments and docstrings are not required except where indicated, although they may help us mark your answers. They may also be worth part marks if you cannot figure out how to complete your code, but completed some part of it.

A mark of at least **40%** (after adjustment, if there is an adjustment) on this exam is required to obtain a passing grade in the course.

MARKING GUIDE

# 1:	_____/15
# 2:	_____/15
# 3:	_____/15
# 4:	_____/ 8
# 5:	_____/ 7
# 6:	_____/12
# 7:	_____/ 4
# 8:	_____/ 8
# 9:	_____/ 6
TOTAL:	_____/90

Good Luck!

OVER...

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 1. [15 MARKS]

Write a function with the signature `most_productive_elf(toys_produced)` which takes in a dictionary `toys_produced` whose keys are names of elves and whose values are the numbers of toys that each elf produced. The function should return the name of the elf who produced the most toys. For example, if `toys0` is `{"Bob":4000, "Gloria":7000, "Hugo":6000, "Grumbles":42}`, `most_productive_elf(toys0)` should return `"Gloria"`. Assume that no two elves produce equal numbers of toys.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 2. [15 MARKS]

Part (a) [12 MARKS]

Write a function with the signature `two_smallest(L)` which takes in a list of `int`s `L`, and returns a list with the two smallest elements of `L`, with the second-smallest element first and the smallest element second. You can assume that all the elements of `L` are different from each other, and that `L` contains more than one element. For example, `two_smallest([5, 3, 10, 4])` should return `[4, 3]`, since `3` and `4` are the smallest elements of `L`, and `4 > 3`. This function can have any runtime complexity you like.

Part (b) [3 MARKS]

What is the runtime complexity of the function you wrote in Part (a)? Explain how you got your answer to this subquestion.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 3. [15 MARKS]

A matrix can be represented using a list of sublists of equal lengths, with each sublist representing a row in the matrix. For example, the matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 0 & 5 & 0 \\ 6 & 7 & 8 & 9 \end{bmatrix}$$

can be represented as:

```
M0 = [ [1, 2, 3, 4],  [5, 0, 5, 0], [6, 7, 8, 9]]
```

Write a function with the signature `largest_col_sum(M)` which takes in a list of lists `M` which represents a matrix in the format outlined above, and returns the sum of the column in the matrix whose sum is the largest. For example, in the matrix `M0` above, the third column has the largest sum: $(3 + 5 + 8 = 16)$. So `largest_col_sum(M0)` should return `16`.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 4. [8 MARKS]

Each of the subquestions in this question contains a piece of code. Treat each piece of code independently (*i.e.*, code in one question is not related to code in another), and **write the expected output for each piece of code**. If the code produces an error, write down the output that the code prints before the error is encountered, and then write "ERROR." You do not have to specify what kind of error it is.

Part (a) [2 MARKS]

```
def f(n):
  n = 5

m = 2
f(m)
print(m)
```

Part (b) [2 MARKS]

```
L = [[1, 2], 3]
M = L[:]
M[0][1] = 5
M[1] = 5
print(L)
```

Part (c) [2 MARKS]

```
def f(d):
  d1 = {}
  for k in d:
    d1[k] = d[k]
  return d1

d = {1:[[1, 2]], 0:[[3, 4]]}
d1 = f(d)
d1[0][0][0] = 5
print(d[0])
```

Part (d) [2 MARKS]

```
s = "HO HO HO"
su = s
s = "Merry Christmas!"
print(su)
```

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 5. [7 MARKS]

The left-hand column in the table below contains different pieces of code that work with list `L`, string `s` and integer `n`. In the right-hand column, give the asymptotic tight upper bound on the worst-case runtime complexity of each piece of code, using Big-Oh notation.

Code	Complexity

`# L is a list with n = len(L)`
`total = 0.0`
`for item in L:`
`  if item > 0.0:`
`  total += item`	

`# L is a list with n = len(L)`
`a = 5`
`for i in range(n):`
`  for j in range(5):`
`  a = i*j`	

`#s is a string of length k,`
`#n is a positive integer`
`def f(s, n, start_str = ""):`
`  if n == 0:`
`  print(start_str)`
`  return`
 	
`  for letter in s:`
`  f(s, n-1, start_str+letter)`
 	
`f(s, n)`	

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 6. [12 MARKS]

Without using `for-` or `while-` loops, write a function with the signature `filter_out_odds(L)` which takes in a list `L` of integers, and returns a new list that contains only the even integers in `L`, in the same order as in `L`. For example, `filter_out_odds([5, -2, 4, 0, 3, 7, 8])` should return `[-2, 4, 0, 8]`. Non-recursive solutions will earn at most 3 marks.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 7. [4 MARKS]

Concisely and precisely state what `f(s, L)` returns for a string `s` and a list of strings `L`. *Suggestion: look at Question 8 before answering this question.*

```
def elem_in_elems(L, e):
    for sub in L:
        if e in sub:
            return True
    return False

def f(s, L):
    #s is a string, L is a list of strings
    res = [s]
    for e in L:
        if not elem_in_elems(res, e):
            continue
        prev_res = res[:]
        res = []
        for sub_res in prev_res:
            split_list = sub_res.split(e)
            for i in range(len(split_list)-1):
                res.extend([split_list[i], e])
            res.append(split_list[-1])

    return res
```

Answer:

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 8. [8 MARKS]

Write a function with the signature `ev(expr)` that takes in a string containing an expression of the form `"<int_1><operation_1><int_2><operation_2><int_3><operation_3>...<int_k>"` and returns the value of that expression. For example,

`ev("10")` should return `10`

`ev("10+2")` should return `12`

`ev("14-3*2")` should return `8`

Assume `expr` is a valid non-empty expression. Assume `<int_1>, <int_2>, ..., <int_k>` can only be positive integers, and the operations can only be `"+"`, `"-"`, or `"*"` (there is no division). Solutions that use `eval` and/or `exec` will not earn any marks.

Hint: the function `f` from Question 7 is actually a present from Santa. You are allowed (but not required) to call it in this question, and it can make your life easier.

For this question **only**, you will earn 1 mark for answering "I don't know." Attempts at a solution where progress towards a correct answer was not made will earn 0 marks.

Use this page for rough work—clearly indicate any section(s) to be marked.

Question 9. [6 MARKS]

Part (a) [3 MARKS]

State clearly and **concisely** what `mystery(L, e)` returns for a list `L` and an integer `e`.

```
def mystery(L, e, i = 0, j = None):
    if j == None or j > len(L):
        j = len(L)

    if i == j:
        return False

    if i == j-1:
        return L[i] == e

    for k in range(i, j, (j-i)//2):
        if mystery(L, e, k, k + (j-i)//2):
            return True

    return False
```

Answer:

Part (b) [3 MARKS]

What is the tight asymptotic upper bound on the worst-case runtime complexity of `mystery(L)` where `n = len(L)`? Use Big-Oh notation. Explain how you got your answer to this subquestion. You may assume that the runtime of `mystery()` is proportional to the number of comparison operations performed.

Use this page for rough work—clearly indicate any section(s) to be marked.

Extra space for solutions

Use this page for rough work—clearly indicate any section(s) to be marked.

Extra space for solutions

PLEASE WRITE NOTHING ON THIS PAGE