CS1010X — Programming Methodology
School of Computing
National University of Singapore

Midterm Test

26 March 2016 **Time allowed:** 1 hour 45 minutes

Student No:	**A**								

Instructions (please read carefully):

1. Write down your matriculation number on the **question paper**. DO NOT WRITE YOUR NAME ON THE QUESTION SET!
2. This is **an open-sheet test**. You are allowed to bring one A4 sheet of notes (written on both sides).
3. This paper comprises **FIVE (5) questions** and **NINETEEN (19) pages**. The time allowed for solving this test is **1 hour 45 minutes**.
4. The maximum score of this test is **100 marks**. The weight of each question is given in square brackets beside the question number.
5. All questions must be answered correctly for the maximum score to be attained.
6. All questions must be answered in the space provided in the answer sheet; no extra sheets will be accepted as answers.
7. The back-sides of the sheets and the pages marked "scratch paper" in the question set may be used as scratch paper.
8. You are allowed to un-staple the sheets while you solve the questions. Please make sure you staple them back in the right order at the end of the test.
9. You are allowed to use pencils, ball-pens or fountain pens, as you like (no red color, please).

GOOD LUCK!

Question	Marks	Remark
Q1		
Q2		
Q3		
Q4		
Q5		
Total		

Question 1: Python Expressions [30 marks]

There are several parts to this problem. Answer each part **independently and separately**. In each part, one or more Python expressions are entered into the interpreter (Python shell). Determine the response printed by the interpreter for the final expression entered and **write the exact output in the answer box**. If the interpreter produces an error message, or enters an infinite loop, explain why. Partial marks may be awarded for workings if the final answer is wrong.

A.

```
x = 75
y = 5
if x%5:
    print("Good")
elif y%75:
    print("No good")
else:
    print("No idea!!")
```

[5 marks]

B.

```
x = 7
y = 3
def a(x):
    return b(x)
def b(y):
    if y == 6:
        return x - y
    return a(x-1)
print(a(4))
```

[5 marks]

C.

```
a = (1,2,3)
b = (5)
c = a + b
for i in c:
    if i == 2:
        print("Cool!")
        break
    elif i%2 ==1:
        print("Beans!")
    else:
        continue
```

[5 marks]

D.

```
a = (1,2,1)
b = (2,1,2)
c = a*2 + b*4 + a + b*3
print(c[3:5] == c[19:21])
```

[5 marks]

E.

```
def compose(f,g):
    return lambda x: f(g(x))
def twice(f):
    return lambda x: f(f(f(x)))
def thrice(f):
    return lambda x: f(f(x))
print(compose(twice,thrice)(lambda x: x+1)(5))
```

[5 marks]

F.

```
def twice(f):
    return lambda x: f(f(f(x)))
def thrice(f):
    return lambda x: f(f(x))
print(twice(thrice)(lambda x: x+3)(2))
```

[5 marks]

Question 2: Recursion & Iteration [21 marks]

Consider the following function `foo`:

$$foo(m,n) = \begin{cases} m & \text{if } m = n, \\ foo(m-n,n) & \text{if } m > n, \\ foo(m,n-m) & \text{otherwise.} \end{cases} \quad (1)$$

A. **[Warm up]** Implement the function `foo` that takes as input two positive integers m and n, which satisfies Equation (1). [4 marks]

```
def foo(m, n):
```

B. What is the order of growth in terms of time and space for the function you wrote in Part (A) in terms of m and n. Explain your answer. [4 marks]

Time:

Space:

C. The following sequence are the list of pairs (m,n) when `foo` is evaluated according to Equation (1) for $m = 171$ and $n = 36$. Basically, it converges in 7 steps. Implement the function `count_steps` that takes as input two positive integers m and n, and returns the number of steps it takes for the function `foo` to converge for m and n. [5 marks]

```
(171, 36)
(135, 36)
(99, 36)
(63, 36)
(27, 36)
(27, 9)
(18, 9)
(9, 9)
```

```
def count_steps(m, n):
```

D. Implement the function `steps` that takes as input two positive integers *m* and *n*, and returns the pairs of steps it takes for the function `foo` to converge for *m* and *n* as a tuple of tuple pairs. **If your answer in Part (C) is a recursive process, implement the function `steps` iteratively. If your answer in Part (C) is an iterative process, implement the function `steps` recursively.** [5 marks]

Example execution:

```
>>> steps(3,2)
((3, 2), (1, 2), (1, 1))

>>> steps(300,200)
((300, 200), (100, 200), (100, 100))

>>> steps(171,31)
((171, 31), (140, 31), (109, 31), (78, 31), (47, 31), (16, 31),
(16, 15), (1, 15), (1, 14), (1, 13), (1, 12), (1, 11), (1, 10),
(1, 9), (1, 8), (1, 7), (1, 6), (1, 5), (1, 4), (1, 3), (1, 2),
(1, 1))

>>> steps(171,36)
((171, 36), (135, 36), (99, 36), (63, 36), (27, 36), (27, 9),
(18, 9), (9, 9))
```

```
def steps(m, n):
```

E. What does `foo` actually compute? [3 marks]

Question 3: Digit Arithmetic [22 marks]

The `digit_sum` of two positive integers a, b is defined as the pairwise sum of their digits modulo 10, i.e. we add each corresponding digit and take the remainder when divided by 10. The following are some worked examples of `digit_sum`:

```
>>> digit_sum(1,2)
3

>>> digit_sum(11,3)
14

>>> digit_sum(55,66)
11

>>> digit_sum(105,13)
118
```

A. Implement the function `digit_sum`. [6 marks]

```
def digit_sum(a,b):
```

B. If your answer in Part (A) is a recursive process, implement the function `digit_sum` iteratively. If your answer in Part (A) is an iterative process, implement the function `digit_sum` recursively. [6 marks]

```
def digit_sum(a,b):
```

The `digit_product` of two positive integers *a*, *b* is defined as the pairwise product of their digits modulo 10, i.e. we multiply each corresponding digit and take the remainder when divided by 10. The following are some worked examples of `digit_product`:

```
>>> digit_product(1,2)
2

>>> digit_product(11,3)
13

>>> digit_product(55,66)
0

>>> digit_product(5,13)
15

>>> digit_product(99,99)
11
```

C. Implement the function `digit_product`. [4 marks]

```
def digit_product(a,b):
```

D. Implement the function `digit_op` that takes in three arguments `a`, `b` and `op` such that we can define `digit_sum` and `digit_product` as follows:

```
def digit_sum(a,b):
    return digit_op(a,b,lambda x,y: x+y)

def digit_product(a,b):
    return digit_op(a,b,lambda x,y: x*y)
```

[6 marks]

```
def digit_op(a,b,op):
```

Question 4: By-Election Fever! [24 marks]

Due to *personal indiscretion* (whatever that means), a by-election will soon be called. Since you have learnt how to code, you have been called for election duty.

Your job is to implement a `polling station` object. What does a polling station do? It accepts votes and counts votes (of course!). It has 4 associated functions:

1. `make_polling_station` returns a new (empty) polling station.
2. `is_polling_station(p)` returns `True` if `p` is a polling station, or `False` otherwise.
3. `add_vote(s, p)` returns a *new* polling station after a new vote for political party `p` is cast at polling station `s`, where `p` is represented with a string.
4. `count_vote(s, p)` returns the number of votes cast at polling station `s` for political party `p`, where `p` is represented with a string. If no votes were ever cast for party `p`, 0 is returned.

Example execution:

```
>>> p = make_polling_station()
>>> is_polling_station(p)
True

>>> is_polling_station("Polling station")
False

>>> p1 = add_vote(p, "PAP")
>>> count_vote(p1,"PAP")
1
>>> count_vote(p1,"SDP")
0

>>> p2 = add_vote(p1, "PAP")
>>> p3 = add_vote(p2, "SDP")
>>> count_vote(p2,"PAP")
2

>>> count_vote(p2,"SDP")
1

>>> p4 = add_vote(p3, "PAP")
>>> count_vote(p4,"PAP")
3

>>> count_vote(p4,"SDP")
1

>>> count_vote(p4,"WP")
0
```

A. Decide on an implementation for the polling station object and implement `make_polling_station`. Describe how the state is stored in your implementation. [4 marks]

Note: You are limited to using **tuples** for this question, i.e. you cannot use lists and other Python data structures.

```
def make_polling_station():
```

B. Implement the function `is_polling_station(p)` that returns `True` if `p` is a polling station, or `False` otherwise. [4 marks]

```
def is_polling_station(s):
```

C. Implement the function `add_vote(s, p)` that returns a *new* polling station after a new vote for political party `p` is cast at polling station `s`. [8 marks]

```
def add_vote(s, p):
```

D. Implement the function `count_vote(s, p)` that returns the number of votes cast at polling station `s` for political party `p`. If no votes were ever cast for party `p`, 0 is returned. [8 marks]

```
def count_vote(s, p):
```

Question 5: Morbid Thoughts [3 marks]

Consider the following thought experiment: you are dead(!) and you are lying in a coffin waiting to be buried. What would you have done in your life so that you would feel that you have not lived your life in vain? Explain.

Appendix

The following are some functions that were introduced in class. For your reference, they are reproduced here.

```
def sum(term, a, next, b):
  if (a > b):
    return 0
  else:
    return term(a) + sum(term, next(a), next, b)

def product(term, a, next, b):
  if a > b:
    return 1
  else:
    return term(a) * product(term, next(a), next, b)

def fold(op, f, n):
  if n==0:
    return f(0)
  else:
    return op(f(n), fold(op, f, n-1))

def enumerate_interval(low, high):
    return tuple(range(low,high+1))

def filter(pred, seq):
    if seq == ():
        return ()
    elif pred(seq[0]):
        return (seq[0],) + filter(pred, seq[1:])
    else:
        return filter(pred, seq[1:])

def accumulate(fn, initial, seq):
    if seq == ():
        return initial
    else:
        return fn(seq[0], accumulate(fn, initial, seq[1:]))
```

Scratch Paper

— E N D O F P A P E R —