CS1010S — Programming Methodology
School of Computing
National University of Singapore

Re-Midterm Test Solutions

20 October 2017 **Time allowed:** 1 hour 30 minutes

Instructions (please read carefully):

1. This is **an open-sheet test**. You are allowed to bring one A4 sheet of notes (written or printed on both sides).
2. The QUESTION SET comprises **FIVE (5) questions** and **TWELVE (12) pages**, and the ANSWER SHEET comprises of **TWELVE (12) pages**.
3. The time allowed for solving this test is **1 hour 30 minutes**.
4. The maximum score of this test is **75 marks**. Your marks will be **capped at 45** if you are taking the test for the second time. 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. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
8. Use of calculators are not allowed in the test.
9. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).

GOOD LUCK!

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: Python Expressions [25 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, **state and explain why**. You may show your workings **outside the answer box**. Partial marks may be awarded for workings even if the final answer is wrong.

A. [5 marks]

```
def f(x):
    return g(y) + x
def g(x):
    return x + y
x = 7
y = 5
print(f(y))
```

B. [5 marks]

```
a = (1, 2, 3)
b = a
a += (a,)
print(a)
print(b)
```

C. [5 marks]

```
j = 0
i = 1
if i > j:
    i = -i
    if j > i:
        j += 1
    elif i < j:
        j += 2
else:
    j += 4
print(i, j)
```

D.

```
y = 1
for x in range(0, 20, 2):
    if (x % y == 0):
        y *= 2
        continue
    elif y > x:
        break
    else:
        y += 1
print(x, y)
```

[5 marks]

E.

```
def f(x):
    return lambda y: x(y)
def g(x, y):
    return lambda z: (x)(y)(z)
print(g(f, f)(g)(lambda x:x+1)(1))
```

[5 marks]

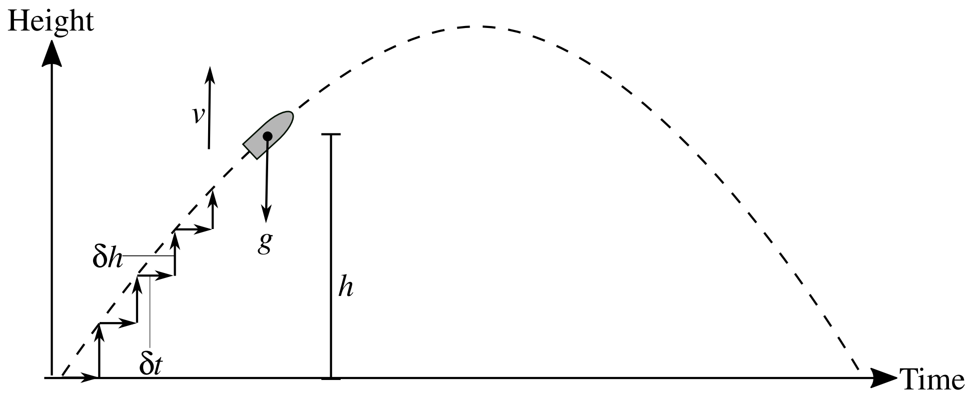
Question 2: Projectile Motion [18 marks]

A common high school physics question is to calculate height and distance of an object thrown in the air. For this question, we will only consider the vertical component of the object, i.e., as if it is thrown directly vertically upwards.

When an object is thrown (or fired) into the air, we assume that the only force acting on it is gravity. In other words, it is constantly accelerating towards the ground. We can then use our high school physics formula to calculate the trajectory.

But once we take into account other factors such as air resistance, variation of gravity, etc. it becomes very complicated to write a formula. That is when we turn to using computers to perform simulations.

In a simulation, we slice time into small discrete intervals, and compute the object's height and velocity at each time interval, as illustrated in the figure below.



For simplicity, we will make the assumptions that gravity is constant and no other forces are acting on the object in flight. Thus, at each time interval δt, the height h of the object increases by its current velocity v, while its velocity decreases by the gravitational acceleration g. In other words, $\delta h = v$ and $\delta v = -g$. Note that when velocity is negative the height will decrease.

We can summarize the calculations using a simple table with some sample parameters:

Time (t)	Height (h)	Velocity (v)	Gravity (g)
0	0	50	9.8
1	50	40.2	9.8
2	90.2	30.4	9.8
3	120.6	20.6	9.8
⋮	⋮	⋮	⋮
10	59.0	-48.0	9.8
11	11.0	-57.8	9.8
12	-46.8	-67.6	9.8

Our table shows that the object returned back to the ground some time between $t = 11$ and $t = 12$. Now the smaller the interval of t, the more precise our simulation will be. But we will not bother with this and leave the precision to be specified by the user by adjusting the units of the parameters.

We can implement a function `height` that takes as input the initial velocity, gravitational acceleration, and time (all real numbers), and returns a tuple of the object's height and velocity at the given time. For example, `height(50, 9.8, 10)` will return `(59.0, -48.0)` (the precision does not matter). If the returned height is negative, it means the object has fallen below its starting height.

A. Provide a recursive implementation of the function `height`. [4 marks]

B. What is the order of growth in terms of time and space for the function you wrote in Part A. Briefly explain your answer. [2 marks]

C. Provide a iterative implementation of the function `height`. [4 marks]

D. What is the order of growth in terms of time and space for the function you wrote in Part (C). Briefly explain your answer. [2 marks]

E. When thrown in the air, an object's height will increase and then decrease following a projectile motion. Using the same method of simulation, we can compute the elapsed time for the object to return pass its starting height, i.e., $h \leq 0$.

Implement the function `flight_time` which takes as input the initial velocity and the gravitational acceleration, and returns the time when the object first returns or falls pass its initial height. [6 marks]

Question 3: Higher-Order Projectiles [12 marks]

A. Consider the higher-order function `fold` which was taught in class.

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

The function `height` in Question 2 can be defined in terms of `fold` as follows:

```
def height(v, g, t):
    <PRE>
    return fold(<T1>, <T2>, <T3>)
```

Please provide possible implementations for the terms T1, T2, and T3. You may optionally define other functions in <PRE> if needed. [4 marks]

B. Gerrie wonders if it is possible to also define the function `flight_time` in terms of the higer-order function `fold`.

Do you think it is possible? If yes, please **provide an implementation**. If no, please **explain in detail why**. [4 marks]

C. Objects thrown at very high velocities often can reach very high altitudes (e.g. an artillery shell). At such altitudes, the gravitational acceleration does not remain constant, but decreases with respect to height based on the following formula:

$$g = \frac{G \times m}{d^2}$$

where m is the mass of the planet, d is the distance from the centre of the planet, and G is the universal gravitational constant 6.67×10^{-11}.

We can modify the `height` function in Question 2 such that it takes in a function that provides the gravitational acceleration at a particular height h above the surface of the earth. The acceleration will be used to compute the velocity instead of a fixed value.

The function `gravity_on_planet` takes as input the mass of the planet and the distance from its centre to its surface. It returns a function that can be used to return the gravitional acceleration at a given height above the planet's surface.

Example:

```
>>> earth_g = gravity_on_planet(5.98e24, 6.38e6) # mass and radius of earth
>>> height(50, earth_g, 10)
(59.05314676331726, -47.98758115155376)

>>> height(400, earth_g, 60)
(6682.95730796391, -186.80108302729832)
```

Provide an implementation of `gravity_on_planet` and the modified `height` function described earlier. [4 marks]

Question 4: Mooncakes [18 marks]

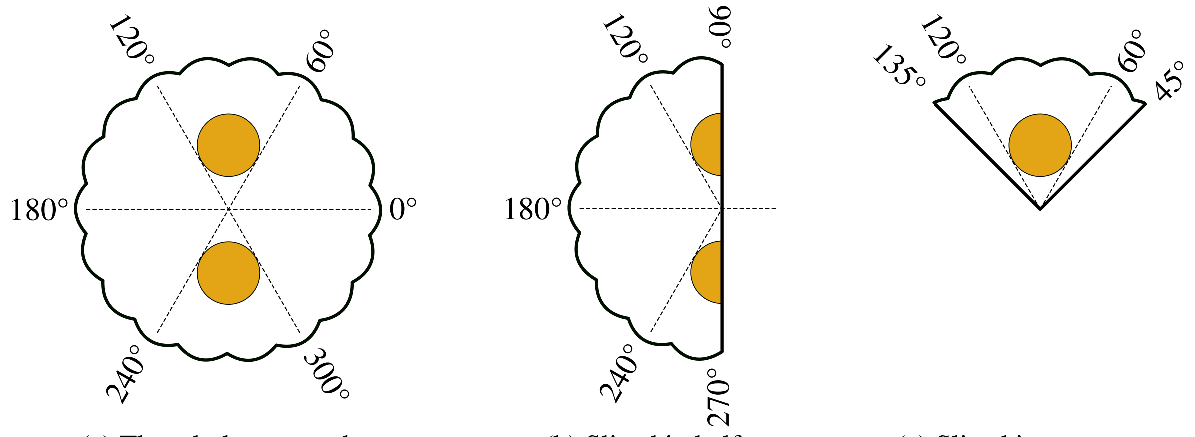
INSTRUCTIONS: Please read the entire question clearly before you attempt this problem!! You are also not to use any Python data types which have not yet been taught in class.

A mooncake is a Chinese bakery product traditionally eaten during the Mid-Autumn Festival. Typical mooncakes are round pastries, measuring about 10 cm in diameter and 3–4 cm thick. A rich thick filling usually made from red bean or lotus seed paste is surrounded by a thin (2–3 mm) crust and may contain yolks from salted duck eggs.

Source: Wikipedia

In this question, we will model a mooncake. Since a mooncake is typically sliced radially into sectors, we can represent a slice of mooncake based on its starting and ending angle. For instance, a complete mooncake will start from 0° to 360°.

Mooncakes can traditionally contain salted yolks or other more modern ingredients. We can model the location of the ingredients in the mooncake by the smallest sector of the mooncake that encompass the ingredient. For example, in the two-yolk mooncake illustrated in Figure 1 has a yolk occupying 60° to 120° and another yolk occupying 240° to 300°.



(a) The whole mooncake. (b) Sliced in half. (c) Sliced in quarter.

Figure 1: Different slices of the same two-yolk mooncake.

A mooncake can be sliced as a sector from a starting angle to an ending angle. For simplicity, we will refer to the sliced piece using the same absolute angles as illustrated in the Figure above.

When a mooncake is sliced, the ingredients within the angles of the sliced sector will remain in the mooncake. For example, when sliced from 90° to 270° (Figure 1b), the mooncake still contains two yolks, but when sliced from 45° to 135° (Figure 1c), only one yolk remains.

Our mooncake model is supported by the following functions:

- `make_mooncake()` takes no inputs and returns a whole mooncake that contains no ingredients.
- `size(mcake)` takes as input a slice of mooncake, and returns the starting angle and ending angle of the slice as a tuple of two elements. Note that a whole mooncake is "sliced" from 0° to 360°.
- `ingredients(mcake)` takes as input a slice of mooncake, and returns a tuple containing each ingredient in the mooncake. Duplicate ingredients will be represented as its own element, e.g. a slice with three yolks will return `("Yolk", "Yolk", "Yolk")`.
- `insert(mcake, ingrd, start, end)` takes as input a slice of mooncake, an ingredient, and a starting and an ending angle (which are numbers), and returns a slice of mooncake with the ingredient added between the stated angles. You may assume that the start and end angles will always fall within the size of the mooncake and that no ingredient crosses the 0°/360° boundary.
- `slice(mcake, start, end)` takes as inputs a mooncake, and a starting and an ending angle (which are numbers) and returns a slice of the mooncake from the starting to ending angle. The ingredients in the resulting slice will remain in the slice if any part of them

fall within the sliced-out angle. You may also assume that the start and end angles will always fall within the size of the mooncake.

Sample run:

```
>>> mooncake = make_mooncake()
>>> mooncake = insert(mooncake, "Yolk", 60, 120)
>>> mooncake = insert(mooncake, "Yolk", 240, 300)
>>> size(mooncake)
(0, 360)

>>> ingredients(mooncake)
("Yolk", "Yolk")  # Two yolks

>>> half = slice(mooncake, 90, 270)
>>> size(half)
(90, 270)

>>> ingredients(half)
("Yolk", "Yolk")  # Still has two separate yolks

>>> q1 = slice(half, 90, 180)
>>> ingredients(q1)
("Yolk",)  # Only one yolk left

>>> q2 = slice(half, 135, 225)
>>> ingredients(q2)
()  # No yolks left
```

To aid you with your implementation, a function `overlap` has been provided that returns the overlapping segment of two segments (x1, y1) and (x2, y2). It is defined as follows:

```
def overlap(x1, y1, x2, y2):
    if x2 > y1 or y2 < x1:
        return ()
    else:
        return (max(x1, x2), min(y1, y2))
```

A. Decide how you will use tuples to model a slice of mooncake, and **draw the box-pointer diagram** corresponding to `mooncake` at the end of the sample run (it is also the mooncake depicted in Figure 1a). [2 marks]

B. Provide an implementation for the functions `make_mooncake`, `size` and `ingredients`. [4 marks]

C. Provide an implementation for the functions `insert` and `slice`. [4 marks]

D. Your friend Jia Zhen recalls a story that mooncakes were once used to store hidden messages and wants try it out to pass messages during lecture. She can easily insert the secret message as an ingredient like so:

```
>>> mooncake = insert(mooncake, "SECRET MSG: Prof is really cool!", 60, 90)
>>> ingredients(mooncake)
("Yolk", "Yolk",  "SECRET MSG: Prof is really cool!")
```

Oh no! 😱 The secret message can easily be seen by anybody who has the mooncake. How embrassing! 😳

Suggest a way, without breaking the abstraction of a mooncake, for Jia Zhen to hide the secret message in the mooncake such that it will not be visible in plain sight, and demonstrate how it works.

In other words, i) show the code to hide her secret message in the mooncake, ii) state what `ingredients(mooncake)` will display, and iii) show the code that someone who knows the technique can read the secret message. If there is absolutely no way to do so in your implementation, explain in detail why. [6 marks]

E. Suppose you were able to solve Jia Zhen's problem and hide secret messages in a mooncake. Jia Zhen is now concerned that if a mooncake is sliced up at a angle that is between the boundaries of a secret message, the message will be cropped off and will not be completely readable.

Do you think Jia Zhen's concern is valid? If so, please **suggest a fix**. If not, please **explain why**. [2 marks]

Question 5: Survey [2 marks]

For your midterm test, you had to write your answers directly in the question paper. For this remidterm test, you are to write your answers in an answer booklet separate from the question paper.

Which format do you prefer, and why? Even if you did not take the first midterm test, you should also give your comments. [2 marks]

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 map(fn, seq):
    if seq == ():
        return ()
    else:
        return (fn(seq[0]),) + map(fn, seq[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

Scratch Paper

— E N D   O F   P A P E R —

CS1010S — Programming Methodology
School of Computing
National University of Singapore

Re-Midterm Test Solutions — Answer Sheet

20 October 2017 **Time allowed:** 1 hour 30 minutes

Student No:	S	O	L	U	T	I	O	N	S

Instructions (please read carefully):

1. Write down your **student number** on this answer sheet. DO NOT WRITE YOUR NAME!
2. This answer sheet comprises **TWELVE (12) pages**.
3. All questions must be answered in the space provided; no extra sheets will be accepted as answers. You may use the extra page at the back if you need more space for your answers.
4. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
5. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).

GOOD LUCK!

For Examiner's Use Only

Question	Marks	Remarks
Q1	/ 25	
Q2	/ 18	
Q3	/ 12	
Q4	/ 18	
Q5	/ 2	
Total	/ 75	

Question 1A

[5 marks]

`15`
This question tests the understanding of scoping.

Question 1B

[5 marks]

`(1, 2, 3, (1, 2, 3))` `(1, 2, 3)`
This question tests the understanding of tuples, and check if the student confuses immutable tuples with mutable lists.

Question 1C

[5 marks]

`20`
This question tests understanding of if-else statements.

Question 1D

[5 marks]

`6 8`

Question 1E

[5 marks]

Error due to incorrect number of inputs. Eventually we will get `f(g)(lambda x:x+1)` which will evaluate to `lambda y: g(y)` which is an error because `g` requires two inputs.

Question 2A [4 marks]

```
def height(v, g, t):
    if time == 0:
        return 0, v
    else:
        h, v = height(v, g, t - 1)
        return h+v, v-g
```

Question 2B [2 marks]

Time: $O(t)$, there is a total of t recursive calls.

Space:$O(t)$, there is a total of t recursive calls, and each call will take up space on the stack.

Question 2C [4 marks]

```
def height(v, g, t):
    h = 0
    for i in range(t):
        h += v
        v -= g
    return h, v
```

Question 2D

[2 marks]

Time: $O(t)$, the loop will iterate n times.

Space:$O(1)$, no extra memory is needed because the variables are overwritten with the new values.

Question 2E

[6 marks]

```
def flight_time(v, g):
    h = 0
    t = 0
    while h > 0 or t == 0:
        t += 1
        h += v
        v -= g
    return t
```

Question 3A [4 marks]

*optional <PRE>:

<T1>:
```
lambda a, b: (b[0] + b[1], b[1] - g)
```

<T2>:
```
lambda x: (0, velocity)
```

<T3>:
```
t
```

Question 3B [4 marks]

No it is not possible. Because in order to use `fold`, we need to specify the value of n which determines how many recursive calls will be done. In other words, we need to know precisely how many computations/recursions have to be made. For `flight_time`, we are trying to compute how many such computations are required, so we do not know the value of n to use.

Question 3C

[4 marks]

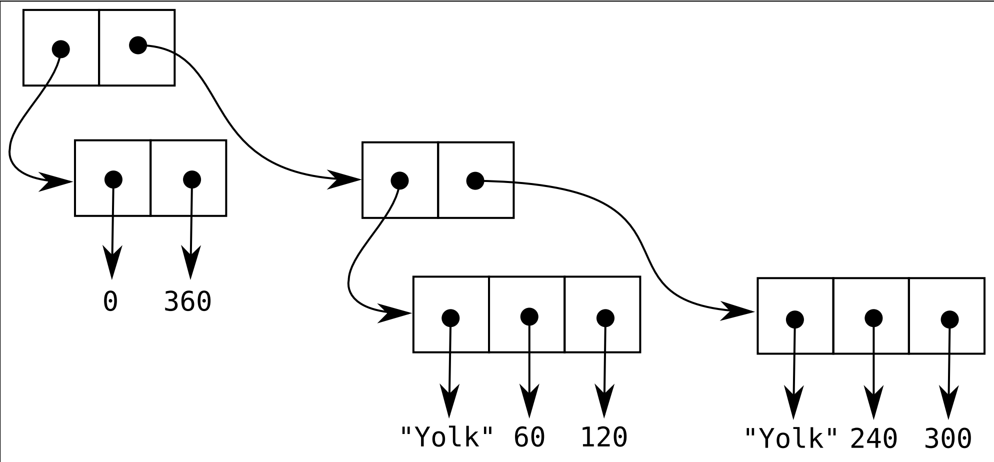
```
def gravity_on_planet(mass, radius):
    return lambda h: 6.67e-11 * mass / (h + raduis)**2




def height(v, g, t):
    h = 0
    for i in range(t):
        h, v = h + v, v - g(h)  # do not use the new h in g(h)
    return h, v
```

Question 4A

[2 marks]



Question 4B

[4 marks]

```
def make_mooncake():
    return ((0, 360), ())




def size(mcake):
    return mcake[0]




def ingredients(mcake):
    return map(lambda x: x[0], mcake[1])  # using map in appendix
```

Question 4C

[4 marks]

```
def insert(mcake, ingrd, start, end):
    piece = (ingrd, start, end)
    return (mcake[0], mcake[1] + (piece,))






# Idea is to use overlap function to quickly check if the ingredient
# should belong in the new slice. There is no need to update the
# ingredient's location
def slice(mcake, start, end):
    ingrds = filter(lambda x: overlap(start, end, x[1], x[2]),
                    mcake[1])
    return ((start, end), ingrds)

# Without using filter
def slice(mcake, start, end):
    ingrds = ()
    for i in mcake[2]:
        if overlap(start, end, i[1], [2]):
            ingrds += (i,)
    return ((start, end), ingrds)
```

Question 4D

[6 marks]

We can hide the secret message in a function like so:

```
>>> msg = lambda : "Prof is really cool\!"
>>> insert(mooncake, msg, 60, 90)
>>> ingrds = ingredients(mooncake)
>>> print(ingrds)
("Yolk", "Yolk", <lambda ... >)
```

This way what is printed will be a lambda function, and not the string itself.

To recover the message, simply do:

```
>>> ingrds[2]()
"Prof is really cool\!"
```

Get the tuple of ingredients and find the hidden message, then call the function, which will return the message.

Other solutions are acceptable as long as they satisfy the requirements of the question.

Question 4E

[2 marks]

No, she has no need to be concerned. Because in our mooncake model, the entire "ingredient" is copied into the new slice. So her secret message will also be copied in its entirety.

Question 5

[2 marks]

Please circle one:

- **Question and Answer in one booklet**
- **Separate Question and Answer booklets**
- **No preference**

Please explain why:

This page is intentionally left blank.
Use it ONLY if you need extra space for your answers, in which case indicate the **question number clearly**.
Do **NOT** use it for your rough work.

This page is intentionally left blank.
Use it ONLY if you need extra space for your answers, in which case indicate the **question number clearly**.
Do **NOT** use it for your rough work.

— END OF ANSWER SHEET —