# NATIONAL UNIVERSITY OF SINGAPORE

## SCHOOL OF COMPUTING

## MID-TERM TEST SOLUTION SKETCHES FOR

## CS1010S Programming Methodology

19 February 2014 Time Allowed: 1 hr 40 min

---

**INSTRUCTIONS TO CANDIDATES**

1. This examination paper consists of **FOUR (4)** questions and comprises **TWELVE (12)** printed pages. Answer **ALL** questions.
2. Read ALL questions before you start.
3. Write your answers in the space provided. If you need more space, write at the back of the sheet containing the question only. DO NOT put part of the answer to one problem on the back of the sheet containing another problem.
4. Please write LEGIBLY! No marks will be given for answers that cannot be deciphered.
5. This is an **OPEN-SHEET** quiz. You are allowed to bring one A4 sheet of notes (written on both sides).
6. Please fill in your **Name** and **Matriculation Number** below.

**Name:** ____________________

**Matriculation Number:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

---

| **For Examiner's Use Only** | | |
|---|---|---|
| | Marks | Max. Marks |
| Question 1 | | **30** |
| Question 2 | | **25** |
| Question 3 | | **25** |
| Question 4 | | **20** |
| **TOTAL:** | | **100** |

## Problem 1 (30 marks)

Answer each part below independently and separately. In each part, one or more Python expressions are entered into the interpreter (Python shell). The shell prompt ">>>" is omitted on each line. Determine the response printed by the interpreter for the final expression entered. If the interpreter produces an error message, or enters an infinite loop, explain why.

a) [5 marks]

```
x = 5
y = 10

for i in range(x):
    x = x + i
    y = y - i

x
```

**Ans: 15**

b) [5 marks]

```
x = 5
y = 10

def add_5(y):
    return lambda: y + 5

add_5(x)()
```

**Ans: 10**

c) [5 marks]

```
x = "loves "
y = "forever"

my_valentine = lambda y: y + x + "me."

my_valentine("Honey ")
```

```
Ans: 'Honey loves me.'
```

d) [5 marks]

```
x = "loves "
y = "forever"
i = 2
j = 3

while j <= 3:
      if i > 0:
          print(x[0:i])
      else: print(y[0:j])
      j = j + 1
      i = i - 1
```

```
Ans: lo
```

e) [5 marks]

```
x = 5
y = 10
z = 15

def f(x):
    def g(y):
        def h(z):
            return x + y + z
        return h(y)
    return g(x)

f(5)
```

**Ans: 15**

f) [5 marks]

```
x = 5
y = 10
z = 15

u = (x, y, z)
u = (x, y, z) + (x, y)

u[0:1]
```

**Ans: (5,)**

## Problem 2 (25 marks)

It's Valentine's Day! Donald wants to buy some roses for his Valentine, Daisy. The roses are sold in bouquets with roses in different sizes. Donald wants to write a Python function `count_roses` that helps him decide how many types of bouquets, with a mix of different sizes of roses, that he can buy for any amount of money. (For example, for X amount of money, he could buy either 1) a bouquet with "A" number of Large roses and "B" number of Small roses, or 2) a bouquet with "C" number of Large roses, "D" number of Medium roses, and "E" number of Small roses; the function `count_roses` would return the result 2 in this case.)

Using wishful thinking, Donald has come up with the following code for the solution:

```
def count_roses(amount, types_of_roses):
    if amount == 0:
        return 1
    elif (amount < 0) or (types_of_roses == 0):
        return 0
    else:
        return <T1> + <T2>
```

Assuming that `types_of_roses` is an integer that indicates the number of types of roses available, and that there is a Python function `most_expensive(<types_of_roses>)` that returns the cost of the most expensive type of roses, give the number of types of roses available.

Help Donald complete the function by providing the code for the expressions **T1** and **T2** above:

a) **T1** indicates the number of ways to count the roses without including the most expensive type of roses [10 marks]:

Grading policy:

Reducing the problem size in this recursive problem can be done in two ways through reducing the input size – reducing the amount, and/or reducing the number of types of roses. Using the "most expensive type" to indicate a type of roses to remove is only a convenient way to reduce the problem size, given the assumption that such a function is available.. Of course other measures can be used in the answers – like least expensive type, or size of the roses, or color of the roses, etc., but then how these measures are related to the amount or cost of the roses need to be defined to establish the correspondence between T1 and T2.

T1

```
count_roses(amount, types_of_roses - 1)
```

b) **T2** indicates the number of ways to count the roses including the most expensive type of roses [10 marks]:

T2:

```
count_roses(amount - most_expensive(types_of_roses), \
            types_of_roses)
```

c) Donald has $10 for the roses that come in three sizes: Large (cost = $4 each), Medium (cost = $2 each), and Small (cost = $1 each). How can he apply the newly defined `count_roses` function to calculate how many types of bouquets he can buy? Write the Python express below [5 marks]:

```
count_roses(10,3)
```

## Problem 3 (25 marks)

PlanetOne, a new telecommunications company in Lion City, is coming up with some promotions for their calling cards. You are tasked to help implement some of these promotions.

a) The “Hello World” international calling card costs (i.e., with a base value of) $10, includes a bonus value of $2 for international phone time and 10 international SMS (the card cannot be used to make local phone calls or send local SMS). A customer buying a new card can include a one-time top-up to the card:

- For a $5 or $10 top-up, you get 50 bonus SMS
- For a $25 top-up, your get $3 of bonus phone time and 100 bonus SMS
- For a $50 top-up, you get $8 of bonus phone time and 200 bonus SMS

Write a Python function `hello_world` that takes in the top-up amount to the card and returns the total phone time value and the number of SMS that the user gets (Hint: use a structured type to return two values at the same time). Only the specified top-up amounts are allowed for this card. [5 marks]

Grading policy:

As long as the assumptions made about the value structure, i.e., what to include or exclude in the phone values and the SMS numbers, and the logic or flow of reasoning are clear in the code, full marks will be given (for this part and also for the rest of the problem).

```
def hello_world(top_up):

    base_value = 10
    base_sms = 10

    if top_up == 0:
        return (base_value + 2, base_sms)
    elif top_up == 5 or top_up == 10:
        return (base_value + 2 + top_up, base_sms + 50)
    elif top_up == 25:
        return (base_value + 3 + top_up, base_sms + 100)
    elif top_up == 50:
        return (base_value + 8 + top_up, base_sms + 200)
    else:
        print("Top up amount unavailable!")
```

b) Due to the lukewarm responses to the international calling cards, PlanetOne is considering introducing a new "Hello All" calling card for both local and international phone calls and SMS. But the company has not decided on the base cost and bonus value, nor the base and bonus SMS to include in the card. A customer buying a new card, however, can include a one-time top-up to the card (where "*" is the multiplication operator):

- For a  $5 or $10 top-up, you get 1 * bonus SMS
- For a $25 top-up, your get 1 * bonus phone time and 2 *  bonus SMS
- For a $50 top-up, you get 2 * bonus phone time and 3 * bonus SMS

Write a Python function `hello_all` that takes in the top-up amount to the card and returns the total phone time charge and number of SMS that the user gets. Only the specified top-up amounts are allowed for this card. State any assumptions you make in your program design [8 marks]

Assumptions and grading policy:

The "undecided" variables  can be globally declared. So they can be changed without affecting the code. However, if other assumptions about their values are clearly stated, full marks will be given as long as the logic in the syntactically code is "reasonable."

```
base_value = 15
base_sms = 15
bonus_value = 5
bonus_sms = 20

def hello_all(top_up):

    if top_up == 0:
        return (base_value + bonus_value, base_sms + bonus_sms)
    elif top_up == 5 or top_up == 10:
        return (base_value + bonus_value + top_up, \
                base_sms + bonus_sms)
    elif top_up == 25:
        return (base_value + 1 * bonus_value + top_up, \
                base_sms + 2 * bonus_sms)
    elif top_up == 50:
        return (base_value + 2 * bonus_value + top_up, \
                base_sms + 3 * bonus_sms)
    else:
        print("Top up amount unavailable!")
```

c) Due to strong competition, PlanetOne is considering introducing a series of new cards that have different top-up bonus schemes for both local and international phone calls and SMS.

Write a Python function `make_card` that:

- "packages" the base value, bonus value, base SMS, and bonus SMS amounts, to be decided at the time of introduction, to a card;
- **instead of** fixing the specific top-up amounts allowed (e.g., 5, 10, 25, 50), also takes in the number of top-up categories (an integer) allowed, and a fixed increment (an integer) for successive categories (e.g., for 3 top-up categories with an increment of 10, the top-up amounts allowed would be 10, 20, 30); and
- includes bonus phone value and bonus SMS for each top-up category that are based on the order of the category in the specified top-up scheme (e.g., for the third top-up category, 3 * bonus value and 3* bonus SMS are included); and
- returns a function that takes in the top-up amount and returns the total phone time charge and number of SMS that the user gets

A template for the `make_card` function is given to you:

```
def make_card \
    (base_value, base_sms, top_up_cats, inc, bonus_value, bonus_sms):
    <Body>
    return <T1>
```

Some examples of using the `make_card` function are shown below:

```
>>> hello_all = make_card(10, 20, 5, 10, 5, 20)
>>> hello_all(80)
<… results returned here …>

>>> hello_valentine = make_card(100, 50, 10, 20, 10, 50)
>>> hello_valentine(180)
<… results returned here …>
```

Fill in the expressions in the **`<Body>`** and the returned results **`<T1>`** in the definition of `make_card` below [12 marks]:

Body:

```
# One possible solution:

def new_card (top_up):
        if top_up == 0:
            return (base_value + bonus_value, base_sms + bonus_sms)
        else:
            for i in range(1, top_up_cats + 1):
                if top_up == i * inc:
                    return (base_value + top_up + bonus_value * i, \
                            base_sms + bonus_sms * i)
        print("Top up amount unavailable!")


# Another elegant solution presented by some students:

def new_card (top_up):
        i = top_up // inc
        j = top_up % inc
        if 0 <= i <= top_up_cats and j == 0:
            return (base_value + top_up + bonus_value* i, \
                    base_sms + bonus_sms * i)
        else:
            print("Top up amount unavailable!")
```

T1:

```
new_card
```

## Problem 4 (20 marks)

For the upcoming school holidays, Resorts Galaxy in Deep Space Nine is coming up with some promotional digital bonus coupons to boost sales of its attractions for the school children. Each coupon type is designed with a specific bonus calculation formula or function that can be readily retrieved from the digital coupons. Each coupon also has a game value `x`, which may be decided at admissions. A customer may hold multiple digital bonus coupons at admissions.

You are asked to help design some programs for use at the attraction entrances to advise on the optimal use of the bonus coupons.

a) Write a Python function `max_two` that takes in two numerical functions `f` and `g` (which indicate the bonus formulas used in two different coupon types respectively), and a positive integer `x` as input, and returns the maximum of applying `f` to `x` and `g` to `x`. You can use the Python primitive operator `max` in your definition. [4 marks]

```
def max_two(f, g, x):

     return max(f(x), g(x))
```

b) To encourage use of the coupons, Spocks wants to offer customers with multiple coupons to enjoy compound benefits, i.e., if a customer has two type '`A`' bonus coupons applied to a game value `x`, the benefits with be `A(A(x))`.

Write a Python function `max_mn` that takes in two numerical functions `f` and `g` (which indicate the bonus formulas used in two different coupons respectively), and three positive integers `m`, `n`, and `x` for the input, and returns the maximum of applying `f` to `x` repeatedly `m` times, and `g` to `x` repeatedly `n` times. You can use the Python primitive operator `max` in your definition. You can (but do not have to) also use the functions `compose`, `identity`, and `repeated` as defined below. [8 marks]:

```
def compose(f, g):
    return lambda x: f(g(x))

def identity(n):
    return n

def repeated(f, n):
    if n == 0:
        return identity
    else:
        return compose(f, repeated(f, n - 1))
```

```
def max_mn(f, g, m, n, x):

    f_m = repeated(f, m)
    g_n = repeated(g, n)
    return max(f_m(x), g_n(x))
```

c) Seeing some good responses, Spocks wants to experiment with different combinations of the promotional digital coupons to provide the best experiences for the customers. Write a Python function `combine` that generalizes the `max_mn` function above to allow the application of any operator `op` (instead of just `max`), as an additional parameter, on the results of repeatedly using the two coupons. ~~You can use the Python primitive operator `max` in your definition.~~ You can (but do not have to) also use the functions `compose`, `identity`, and `repeated` as defined above in part b) [8 marks]

```
def combine(f, g, m, n, op, x):

    f_m = repeated(f, m)
    g_n = repeated(g, n)
    return op(f_m(x), g_n(x))
```

*** END OF PAPER ***