CS1010X — Programming Methodology
School of Computing
National University of Singapore

Midterm Test

27 March 2021 **Time allowed:** 2 hours

Student No: | | | | | | | | | |

Instructions (please read carefully):

1. Write down your student number on the **question paper and all pages of the provided answer sheets**. DO NOT WRITE YOUR NAME ON THE QUESTION PAPER AND ANSWER SHEETS !
2. This is **an open-book test**.
3. Please switch off your mobile phone, and no laptop and no calculator. If found using any of these equipment any moment from now till your scripts have been collected, you will be given a zero mark.
4. This paper comprises **FOUR (4) questions** and **ELEVEN (11) pages**. The time allowed for solving this test is **2 hours**.
5. The maximum score of this test is **32 marks**. The weight of each question is given in square brackets beside the question number.
6. All questions must be answered in the space provided in the answer sheets (not this question paper). At the end of the test, please submit both your question paper and your answer sheets. No extra sheets will be accepted as answers.
7. You are allowed to use pencils. Please be sure your handwriting is legible, and you have proper indentation as needed for your Python codes.

GOOD LUCK!

Question	Marks
Q1	
Q2	
Q3	
Q4	
Total	

Question 1: Warm up - fill in the blank [1 marks]

Our (Python) computer program consists of 3 main constructs: sequential statements, _____ and _____ statements.

Question 2: Python Expressions [9 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 all the response printed by the interpreter for the expressions entered and **write the exact output in the answer box**.

It is assumed that there is no syntax errors in our input to the Python interpreter. If the interpreter produces an execution error message at some stage, state where and the nature of the error (you don't need to show the exact error message but still need to write out any output from the Python expressions before the error was encountered). In another extreme, if the interpreter enters an infinite loop, explain how it happens. **Partial marks may be awarded for working if your answer is wrong. You are, however, responsible to make clear the part of your answer verse the part of your working.**

A. [2 marks]

```
def f(x):
    if x == 0:
        print("ready")
        return y
    elif x == 1 or y == 0:
        print("relax...")
    elif y == 1:
        print("go")
    x = x - 1
    f(x)
    return False

x = 2
y = 1
y = f(y)
print(not f(y+x))
```

Answer:

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

B. Below is a single statement entered into Python shell. [1 marks]

```
120,000 > 1,000
```

Answer:

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

C. [2 marks]

```
def f(x):
    if x==0:
        return 0
    result = 0
    for x in range(20,0,-2):
        x = 10
        result += x
    print (result)

x=3
y=1
x = f(x)
print (not x)
```

Answer:

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

D. Below are 8 statements entered into Python shell. [2 marks]

```
word = "01010101010101010101" # len(word) = 22
mask = "-------*------*-------" # 7 "-" then "*" then 6 "-" then "*" then 7 "-"
masking = ( (11,-11), (6, -6), (3, -3), (2, -2), (1, -1) )

for i in range (0, 5):
    word = word[0::2]+word[1::2]
    maskedWord = word[0:masking[i][0]] + mask[masking[i][0]: masking[i][1]] + \
                 word[masking[i][1]:]
    print(maskedWord)
```

Answer: (please align your output pattern nicely in a grid to avoid penalty)

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

E. [2 marks]

```
def comp(f, g):
    return lambda g: f(f(g))

def add2(x):
    return x+2

def add3(x):
    return lambda x: x+3

print ( add3(add2)(8) )
print ( add3((add2)(8))(88) )
print ( comp(add3, add2)(8)(88) )
print ( comp( add3(add2), add3(add2) )(8))
```

Answer:

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

Question 3: Iteration & Recursion, with Time & Space [10 marks]

Suppose you are given the following functions:

```
def A(n):
    if n == 1:
        return 1
    else:
        return n + B(n+1)

def B(n):
    if n == 1:
        return 1
    else:
        return n + A(n//2)
```

A. What is the output value of `A(50)`? [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

B. Provide the time and space complexity of the function `A` in terms of input `n`. Justify your answer with consideration of the details of the given programs. [3 marks]

Time:
Space:
Justification (for time and for space):

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

C. Provide an iterative implementation of `A` (using `while` loop) with an input n without calling any other functions (such as `B`). Note that the output value from your program MUST be the same as that given by `A` for the same input n. [3 marks]

```
def A_itr ( n ):
```

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

D. Suppose we now want to use the functions `A` and `B` for tuples. We amended them to become `A_tuple` and `B_tuple` as follows:

```
def A_tuple(t):
    if len(t) == 1:
        return t[0]
    else:
        return (lambda t: t[0] if len(t)>0 else 0)(t) + B_tuple(t + (len(t),))

def B_tuple(t):
    if len(t) == 1:
        return t[0]
    else:
        return (lambda t: t[0] if len(t)>0 else 0)(t) + A_tuple(t[0:len(t)//2])
```

Provide the time and space complexity of the function `A_tuple` in terms of the length n of the input tuple. Justify your answer. [3 marks]

Time:
Space:
Justification (for time and for space):

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

Question 4: Higher Order Functions [12 marks]

You will be working with the following higher-order function `sumT` which is doing sum of the elements of a tuple `t` with `term(t)`:

```
def sumT(t, term, next):
    if t == ():
        return ()
    else:
        return term(t) + sumT( next(t), term, next )
```

In the following, we are going to use `sumT` to help to perform operations with just one tuple, two tuples, three tuples and tuple of tuples.

A. The function `prefix_sum` takes an input tuple `t` (of numbers) to compute the prefix sum of each term in `t`. That is, the output is another tuple whose i^{th} element is the sum of all elements of `t` with index 0 till index i:

Example execution:

```
>>> prefix_sum( (1,2,3,4,5,6,7,8,9) )
(1, 3, 6, 10, 15, 21, 28, 36, 45)

>>> prefix_sum( (1,2,3,4,5,6,7,8,9,10) )
(1, 3, 6, 10, 15, 21, 28, 36, 45, 55)
```

We can define `prefix_sum` with `sumT` as follows:

```
def prefix_sum( t ):
    return sumT( t, <T1>, <T2> )
```

Please provide possible lambda functions for `<T1>` and `<T2>`. [2 marks]

`<T1>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

`<T2>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

B. The function `interleave` takes two tuples as input to output a tuple whose elements are alternately taken from the two tuples (while maintaining the same order as in the input tuples). We assume the two input tuples are of the same length.

Example execution:

```
>>> interleave( (1,2,3), ("a", "b", "c") )
(1, 'a', 2, 'b', 3, 'c')

>>> interleave( (1,2,3,4), ("a", "b", "c", (1,2) ) )
(1, 'a', 2, 'b', 3, 'c', 4, (1, 2))
```

We can define `interleave` with `sumT` as follows:

```
def interleave(t1, t2):
      return sumT( t1 + t2, <T3>, <T4> )
```

Please provide possible lambda functions for `<T3>` and `<T4>`. [2 marks]

`<T3>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

`<T4>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

C. The function `average` takes three input tuples (of numbers) to compute the average of the three corresponding elements of the three tuples. We assume the three input tuples are of the same length.

Example execution:

```
>>> average( (3, 4, 5), (1, 2, 3), (2, 9, 4) )
(2.0, 5.0, 4.0)

>>> average( (3, 4, 5, 6), (1, 2, 3, 4), (2, 9, 4, 8))
(2.0, 5.0, 4.0, 6.0)
```

We can define `average` with `sumT` as follows:

```
def average(t1, t2, t3):
      return sumT( (t1, t2, t3), <T5>, <T6> )
```

Please provide possible lambda functions for `<T5>` and `<T6>`. [2 marks]

`<T5>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

`<T6>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

D. We have the following `map` function in our lecture.

```
def map(f, t):
    if t == ():
        return ()
    else:
        return (f(t[0]),) + map(f, t[1:])
```

We can also define `map` with `sumT` as follows:

```
def map(f, t):
      return sumT( t, lambda t: (f(t[0]),), lambda t: t[1:] )
```

For this question, we assume we use the `map` given here in terms of `sumT`. We can now compute the `transpose` for a matrix M given as a tuple of tuples. We have $M[0]$ as the row 1 of the matrix, and $M[1]$ of row 2, and, in general, $M[i]$ as the row i of the matrix. And, $M[i][j]$ is the element at row i and column j. The transpose of a matrix M with n rows and m columns is another matrix M^T with m rows and n columns with its element $M^T[j][i] = M[i][j]$.

Example execution:

```
>>> # below is a matrix M with 3 rows and 3 columns where
>>> # the first row is (1,2,3), second (4,5,6), and third (7,8,9)
>>> # the first column is (1,4,7), second (2,5,8), and third (3,6,9)
>>> M = ( (1,2,3), (4,5,6), (7,8,9) )
>>> transpose(M)
((1, 4, 7), (2, 5, 8), (3, 6, 9))
>>> # the transpose of M has 3 rows and 3 columns,
>>> # with the first row of the transpose taken from M's first column, etc.

>>> N = ( (1,2,3,4), (5,6,7,8), (9,10,11,12) ) # has 3 rows and 4 columns
>>> transpose(N)
((1, 5, 9), (2, 6, 10), (3, 7, 11), (4, 8, 12))
>>> # the transpose of N has 4 rows and 3 columns
>>> # with the first row of the transpose taken from N's first column, etc.
```

We can also define `transpose` with `sumT` as follows:

```
def transpose ( M ):
      return sumT( M, <T7>, <T8> )
```

Please provide possible lambda functions for `<T7>` and `<T8>` (that involves `map` that uses `sumT` as given here and not the one in the lecture note). [2 marks]

`<T7>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

`<T8>`: [1 marks]

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

E. Among the above solutions of yours for `prefix_sum`, `interleave`, and `average`, pick any two of them to discuss their time and space complexity. Note that your answers given in the previous parts must be almost correct before we can grade your given time and space complexity here. So, please choose carefully the two that you are most confident of their correctness to fill in their corresponding two boxes (and leave one box empty). Note that if all 3 boxes are filled, we will grade just the first two. For each box, please state first the time and space complexity, and then provide your reasoning - note that all three may seem to be the same, you still need to point our their specific details (or subtle differences) to receive full credit. [2 marks]

`prefix_sum:`

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

`interleave:`

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

`average:`

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

F. Analyse the time and space complexity of `transpose` in terms of n and m where n is the number of rows and m is the number of columns of the input matrix. Note that your answers given to `transpose` in the above must be almost correct before we can grade your attempt here. [2 marks]

`transpose`:

Space:

Justification (for time and for space):

* PLEASE WRITE YOUR ANSWER IN THE ANSWER SHEETS – here could be used mainly for your draft *

— END OF PAPER —