Programming Tasks for Practical Examination

CS1101S — Programming Methodology
National University of Singapore
School of Computing

Semester 1 AY2014/2015

11 November 2014 Time Allowed: **1 Hour 45 Minutes**

Instructions (please read carefully):

1. This question booklet comprises **10 pages** and has **3 questions** with a total of **64 marks**. Answer all questions.

2. This is an **open book** exam. Any written or printed material, or programs stored on the JFDI Academy may be used as reference material.

3. The internet must not be used, except the site http://jedi.ddns.comp.nus.edu.sg.

4. The questions in this paper should be answered and submitted on the JFDI Academy at http://jedi.ddns.comp.nus.edu.sg. The questions are listed as "Padawan Exam 2014 Q1" to "Padawan Exam 2014 Q3".

5. Remember to **finalize** your submissions at the end of the practical exam. However, remember that submissions **cannot be edited** after they have been finalized.

6. The `assert` function calls in the solution templates provide some sample test cases to check the correctness of your solutions. You may view the test results in the 'Display' tab. Note that the given test cases are not exhaustive; passing them does not mean your solution is correct. You are strongly encouraged to add your own test cases.

7. The solutions of some sub-questions require correct solutions of previous questions and sub-questions. In the environment that we set up for you, **you can program and test your solutions without dependencies**. This is achieved by the `assert` function. Each time `assert` is called, our correct implementation of the solutions to all relevant previous questions and sub-questions is installed in the global environment. We therefore strongly encourage you to **test your programs only using** `assert`.

GOOD LUCK!

Question 1: Matriculation Numbers [15 marks]

We would like to have a program to map a ***Student NUSNET ID***, such as u0901234 and a0123456, to its corresponding ***matriculation number***, such as U091234H and A0123456J.

The last alphabet letter in a matriculation number is called the ***check digit***, and it is computed from the numeric digits in the matriculation number.

There are two types of IDs and matriculation numbers — U-prefixed and A-prefixed:

- To convert a U-prefixed ID to a matriculation number, the "u" prefix is converted to a "U", the third numeric digit (counting from left) is discarded, and the remaining numeric digits are used to compute a check digit.
- To convert an A-prefixed ID to a matriculation number, the "a" prefix is converted to an "A", and the numeric digits are used to compute a check digit.

The check digit is computed as follows. Let d_1 to d_6 (for U-prefixed) or d_1 to d_7 (for A-prefixed) be the numeric digits (from left to right) in the matriculation number. Compute the weighted sum $s = w_1 \times d_1 + ... + w_6 \times d_6$ or $s = w_1 \times d_1 + ... + w_7 \times d_7$ using the following weights:

Prefix	w_1	w_2	w_3	w_4	w_5	w_6	w_7
U	0	1	3	1	2	7	
A	1	1	1	1	1	1	1

Finally, from the following table, find the check digit that corresponds to the remainder of the weighted sum s divided by 13 (s modulo 13).

Remainder	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Check Digit	Y	X	W	U	R	N	M	L	J	H	E	A	B

A. [3 marks]

Write a function `weighted_sum(digits, weights)` that takes in a list of digits and a list of weights and returns the weighted sum of the digits. You can assume that `digits` and `weights` are the same length.

Examples:

```
weighted_sum(list(2, 3), list(5, 4));    // returns 22

weighted_sum(list(1, 2), list(0, 2));    // returns 4
```

B. [4 marks]

Write a function `discard_element(xs, pos)` that takes in a list `xs` and a position `pos` and returns a copy of `xs` with the element at position `pos` discarded. Note that the first element in a list has position 0.

Examples:

```
discard_element(list(1, 2, 3), 0);
// returns list(2, 3)

discard_element(list(1, 2, 3), 2);
// returns list(1, 2)

discard_element(list(1, 2, 3), 3);
// returns list(1, 2, 3)
```

C. [8 marks]

Write a function `id_to_matric(id)` that takes in a list `id` that represents a student NUSNET ID and returns a list that represents the matriculation number that corresponds to the input ID. You can assume that the inputs are always valid.

Examples:

```
id_to_matric(list("u", 0, 9, 0, 1, 2, 3, 4));
// returns list("U", 0, 9, 1, 2, 3, 4, "H")

id_to_matric(list("a", 0, 1, 2, 3, 4, 5, 6));
// returns list("A", 0, 1, 2, 3, 4, 5, 6, "J")
```

Question 2: Shipping Books [25 marks]

You are running an online book store. Each day at the warehouse, you will pack books ordered by each customer into one or more cuboid-shaped packages and send them to your customers via a shipping company.

Each book is also cuboid-shaped, with length, width, and height, all measured in meters (m). The length runs along the book spine, and the height is the thickness of the book. The width may be longer than the length for some books. The mass of the book is measured in kilograms (kg).

You have a ***standard packing method*** to pack a set of books into a package. You always stack the books cover-to-cover, and with all the spines parallel to each other. Then you will make a packaging box with height exactly the same as the total height (thickness) of the books. The length of the box will be equal to the largest length of the books, and the width will be equal to the largest width of the books. The package's actual mass is the total mass of the books in it.

Your shipping company is using the concept of "*volumetric mass*" to charge you. For each package, the ***billable mass*** is the higher of the following two:

- the volume of the package in m^3 times 500 kg/m^3, or
- the actual mass of the package.

The ***shipping charge*** for each package is calculated as follows:

1) The billable mass is rounded up to the nearest whole kg.
2) For the first kg — $1.00.
3) For the second kg — $0.80.
4) For the third kg — $0.60.
5) Subsequent kg — $0.40 per kg.

For example, for a package with billable mass of 4.5 kg, the shipping charge will be $1.00 + $0.80 + $0.60 + 2 × $0.40 = $3.20.

For the following tasks, the `CuboidObject` class has been defined for you. A `CuboidObject` object has the fields `length`, `width`, `height` and `mass`. It has methods `get_length`, `get_width`, `get_height`, `get_mass`, `set_length`, `set_width`, `set_height`, `set_mass`, and `get_volume`. All length quantities are in meters, all mass quantities in kg, and volume in m^3.

For the entire question, you must not modify the existing `CuboidObject` class definition.

A. [2 marks]

Define the `Book` class as a subclass of `CuboidObject`. A `Book` object has an additional field, `title`, and the methods `get_title()` and `set_title(title)`. The constructor takes a title, length, width, height (all in meters) and mass (in kg) as arguments.

Example:

```
var book = new Book("Myths", 0.30, 0.20, 0.04, 1.5);
book.get_title();     // returns "Myths"
book.get_length();    // returns 0.3
book.get_width();     // returns 0.2
book.get_height();    // returns 0.04
book.get_mass();      // returns 1.5
book.set_title("More Myths");
book.get_title();     // returns "More Myths"
```

B. [3 marks]

Write a function `books_total_height(books)` that takes in a list of book objects and returns the total height (thickness) of the books.

Write a function `books_total_mass(books)` that takes in a list of book objects and returns the total mass of the books.

You get full marks for this task only if you use the `accumulate` function in a correct and meaningful way in the above two functions.

Example:

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 1.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 2.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 1.2);
var books = list(book1, book2, book3);
books_total_height(books);    // returns 0.12
books_total_mass(books);      // returns 4.7
```

C. [4 marks]

Write a function `package_length_width(books)` that takes in a list of book objects and returns a `pair` that contains the length and width, respectively, of the package produced using the standard packing method described above. You get full marks for this task only if you use the `accumulate` function in a correct and meaningful way.

Example:

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 1.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 2.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 1.2);
var books = list(book1, book2, book3);
package_length_width(books);  // returns pair(0.35, 0.3)
```

D. [4 marks]

Define the `Package` class as a subclass of `CuboidObject`. A `Package` object has an additional field, `books`, which is the list of book objects placed in the package. It has an additional method `get_books()` that returns the list of book objects in the package. The constructor takes in a list of book objects, and finds out the length, width, height, and mass of the package using the standard packing method described above.

Example:

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 1.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 2.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 1.2);
var books = list(book1, book2, book3);
var package = new Package(books);
package.get_books();     // returns books
package.get_length();    // returns 0.35
package.get_width();     // returns 0.3
package.get_height();    // returns 0.12
package.get_mass();      // returns 4.7
```

E. [2 marks]

Define a method, `get_billable_mass()`, for a `Package` object, to return the billable mass of the package.

Examples:

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 1.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 2.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 1.2);
var books = list(book1, book2, book3);
var package = new Package(books);
package.get_billable_mass();   // returns 6.3


var book1 = new Book("Book1", 0.30, 0.20, 0.04, 2.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 3.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 2.2);
var books = list(book1, book2, book3);
var package = new Package(books);
package.get_billable_mass();   // returns 7.7
```

F. [4 marks]

Define a method, `get_shipping_charge()`, for a `Package` object, to return the shipping charge of the package.

Examples:

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 1.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 2.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 1.2);
var books = list(book1, book2, book3);
var package = new Package(books);
package.get_shipping_charge();   // returns 4
```

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 2.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 3.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 2.2);
var books = list(book1, book2, book3);
var package = new Package(books);
package.get_shipping_charge();   // returns 4.4
```

G. [6 marks]

Given a set of books to be put in a package, the standard packing method described above often does not produce the minimum package volume. We want to minimize the package volume by allowing some books, if necessary, to be rotated 90 degrees such that their spines run along the width of the packaging box.

Write a function `improved_package_length_width(books)` that takes in a list of book objects and returns a `pair` that contains the length and width, respectively, of the package that has the minimum volume that can be produced using the improved packing method. The returned length must be larger or equal to the width.

Example:

```
var book1 = new Book("Book1", 0.30, 0.20, 0.04, 1.5);
var book2 = new Book("Book2", 0.25, 0.30, 0.05, 2.0);
var book3 = new Book("Book3", 0.35, 0.20, 0.03, 1.2);
var books = list(book1, book2, book3);

improved_package_length_width(books);
// returns pair(0.35, 0.25)
```

Question 3: Matrices [24 marks]

You are to complete a `Matrix` class definition by adding methods that operate on matrices. A `Matrix` object internally represents an $m \times n$ matrix as a list of m lists, where each of those m lists has n numbers. For example, a 2×3 matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is represented as `list(list(1, 2, 3), list(4, 5, 6))`.

The `Matrix` constructor has been defined for you. It takes in the arguments `elems`, `num_rows`, and `num_cols`. Argument `elems` is a list of lists of numbers that represents the input matrix. Arguments `num_rows` and `num_cols` are the number of rows and columns the input matrix has. You can assume that `num_rows` and `num_cols` match the actual number of rows and columns in `elems`. If `elems` is an empty list, then a zero matrix of size `num_rows` × `num_cols` is created in the `Matrix` object.

Note that the top row of a matrix is Row 1, and the left-most column is Column 1.

For the entire question, you must not modify the existing `Matrix` class definition.

<u>IMPORTANT</u>
You are not allowed to use `for` or `while` loops for this task. No credit will be awarded for solutions using `for` or `while` loops in this task.

A. [2 mark]

Define a method, `get_elem(row, col)`, for a `Matrix` object, to return the value of the matrix element at row `row` and column `col`. You can assume the input `row` and `col` are always valid. Please be reminded that the top row of a matrix is Row 1, and the left-most column is Column 1.

Examples:

```
var mat = new Matrix(
                list(list(1, 2, 3), list(4, 5, 6)), 2, 3);
mat.get_elem(1, 2);    // returns 2
mat.get_elem(2, 1);    // returns 4
```

B. [4 mark]

Define a method, `set_elem(row, col, new_val)`, for a `Matrix` object, to set the value of the matrix element at row `row` and column `col` to `new_val`. You must use the `set_head` or `set_tail` function to modify the matrix element.

Example:

```
var mat = new Matrix(
                list(list(1, 2, 3), list(4, 5, 6)), 2, 3);
mat.get_elem(2, 3);   // returns 6
mat.set_elem(2, 3, 9);
mat.get_elem(2, 3);   // returns 9
```

C. [6 marks]

Define a method, `scale(k)`, for a `Matrix` object `mat`, such that `mat.scale(k)` returns a new `Matrix` object whose matrix elements are `k` times the corresponding matrix elements in `mat`. You get full marks for this task only if you use the `map` function in a correct and meaningful way.

Example:

```
var mat = new Matrix(
                list(list(1, 2, 3), list(4, 5, 6)), 2, 3);
var mat2 = mat.scale(2);

mat2.get_all_elems();
// returns list(list(2, 4, 6), list(8, 10, 12))

mat.get_all_elems();
// returns list(list(1, 2, 3), list(4, 5, 6))
```

D. [6 marks]

Let matrix ***B*** be the ***transpose*** of matrix ***A***, then every element $a_{r,c}$ of ***A*** is equal to $b_{c,r}$ of ***B***.
For example, the transpose of the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is the matrix $\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$.

Define a method, `transpose()`, for a `Matrix` object `mat`, such that `mat.transpose()` returns a new `Matrix` object whose matrix is the transpose of that of `mat`.

Example:

```
var mat = new Matrix(
                list(list(1, 2, 3), list(4, 5, 6)), 2, 3);
var mat2 = mat.transpose();

mat2.get_all_elems();
// returns list(list(1, 4), list(2, 5), list(3, 6))

mat.get_all_elems();
// returns list(list(1, 2, 3), list(4, 5, 6))
```

E. [6 marks]

Let ***A*** be a matrix of size $m \times p$, and ***B*** be a matrix of size $p \times n$. If ***C*** is the ***product*** of ***A*** and ***B***, then ***C*** is a matrix of size $m \times n$, where each element of ***C***, $c_{i,j} = \sum_{k=1}^{p} a_{i,k} \cdot b_{k,j}$.

Define a method, `multiply(mat2)`, for a `Matrix` object `mat`, such that `mat.multiply(mat2)` returns a new `Matrix` object whose matrix is the product of `mat` and `mat2`. You can assume that the matrix sizes of `mat` and `mat2` are compatible for multiplication.

Example:

```
var mat = new Matrix(
                list(list(1, 2, 3), list(4, 5, 6)), 2, 3);
var mat2 = new Matrix(
                list(list(12, 9), list(11, 8), list(10, 7)),
                3, 2);
var mat3 = mat.multiply(mat2);

mat3.get_all_elems();
// returns list(list(64, 46), list(163, 118))
```

——— END OF PAPER ———