NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Mock Practical Examination 1 (PE1) for Semester 2, AY2011/2
CS1010 — Programming Methodology

13/14/15 February 2012 Time Allowed: 1½ hour

INSTRUCTION TO CANDIDATES

1. This paper consists of **2** exercises on **3** pages. Each exercise is 50%.
2. This is an open-book test. You may bring in any printed material, but **not** electronic devices, including but not limited to laptops, thumb-drives and electronic dictionaries. Please switch off / silence your mobile phone and keep it out of view.
3. Please login to your own UNIX account for programming. (**In real PE, you will be given a special account**.)
4. No skeleton programs are provided for mock PE. Please write everything from scratch. (**In real PE, you will be given skeleton programs**.)
5. At the end of the mock PE, submit your programs to CodeCrunch. You only have **one** attempt for your submission! (**In real PE, you will not submit your programs to CodeCrunch. We will collect your source codes after PE**.)
6. Please read carefully and follow all instructions in the questions. If in doubt, please ask. Raise your hand and the invigilator will attend to you.
7. You are **not** allowed to use any C language syntax not covered in class (e.g., arrays or string functions). If in doubt, please check with the invigilator.
8. You may assume that all inputs are valid, that is, you do not need to perform input validity check.
9. Please save your programs regularly during the mock PE.

ALL THE BEST!

Exercise 1: Inversion [50 marks]

Problem Statement

Write a program **inversion.c** that reads a positive integer `num` and processes the following:

1. Find the reverse value of `num`, e.g., 12 -> 21.
2. If `num` does not equal to its reverse value, update `num` with the absolute difference between `num` and its reverse value, e.g., |12-21| = 9
3. Repeat step 1-2 until `num` equals its reverse value. Count the number of iterations it takes to reach such a `num`.

Modular design is required:

- Write a function `reverse` for step 1. You are to determine its return type and formal parameter(s) by yourself.
- Write a function `count_round` to calculate the number of iterations it takes to finish repetition in step 3. Print out the number of iterations needed in the main function. Again, you are to determine its return type and formal parameter(s) by yourself.

Check sample runs for input and output format.

Sample runs

Four sample runs are shown below with user input highlighted in **bold**.

```
Enter a positive number: 9
Number of rounds needed: 1
```

```
Enter a positive number: 12
Number of rounds needed: 2
```

```
Enter a positive number: 321
Number of rounds needed: 6
```

```
Enter a positive number: 1234
Number of rounds needed: 11
```

Exercise 2: Analysis [50 marks]

Problem Statement

The second exercise is a continuation of the first exercise. Please make a copy of your Exercise 1 program and name it as **analysis.c**.

1. Here your `count_round` function should return the final value of `num` rather than the number of iterations taken (therefore please rename this function as `palindrome`).
2. If the final value of `num`:
    - is of 1 or 3 digits, print out "Bingo!"
    - is of 2 digits, print out "Ding dong!"
    - otherwise, print out "Incredible"

You are to write a separate function `analyse` to analyse (using `switch` statement) and print out corresponding messages. You are to determine its return type and formal parameter(s) by yourself.

You may write more helper functions if needed.

Check sample runs for input and output format.

Sample runs

Three sample runs are shown below with user input highlighted in **bold**.

```
Enter a positive number: 12
Bingo!
```

```
Enter a positive number: 321
Ding dong!
```

```
Enter a positive number: 1221
Incredible!
```

=== END OF PAPER ===