School of Computer Science
Systems and Architecture (COMP1040) Autumn 2018

Assembly Programming Coursework
Deadline: Monday 26th of November, 2018

Collaborating in small groups of up to three students is permitted, but you must implement your own programs (absolutely *do not* copy and paste from others) and provide your own answers where appropriate.

Note that lacking proper comments and user prompts will lose mark.

Submit your UNCOMPRESSED assembly program files to Moodle.

1. Write a program in MIPS32 assembly language which reads a positive integer N and prints out the following:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
...
1 2 3 4 5 6 ... N
```

(15 marks)

2. Implement a program which prompts user two integers inputs x, y from the console and calculate the following expression in signed 32-bit arithmetic:

$$x^3 + 3x^2y + 3xy^2 + 9y^3$$

Note that you are NOT allowed to use pseudo-instructions with overflow checking for the calculation (i.e. you can not use `mulo`). If an overflow occurs during any step of the calculation, you should print an error message instead, and stop the program.

Hint: You could simplify the expression before calculation. Please remember to test your program with a range of different inputs, e.g. $x = 2, y = 3$; $x = -3, y = 4$; $x = 1\,000, y = 150\,000\ldots$

(25 marks)

3. Write a program in MIPS32 assembly language which takes three arguments: register `A` will receive the initial address of a string, register `B` will receive a character and register `C` will receive another character. Within the string, your program should replace any occurrence of the char stored in `B` by the char stored in register `C`. For example:

```
Input string: Apple
Input character: p
Input character: q
Output: Aqqle
```

(25 marks)

4. The Newton-Raphson method for calculating the *positive* square root of a number $n \geq 0$ is given as:

$$x_0 \approx \sqrt{n}$$
$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{n}{x_i}\right)$$

with $\sqrt{n} = \lim_{i \to \infty} x_i$.

Since we are working with a finite representation of floating point numbers, we can stop when the difference between x_i and x_{i+1} is small enough. The following pseudo-C program illustrates the Newton-Raphson method:

```
// input a floating point number n
float x0 = n, x1 = 0.5 * n;
while(abs(x0 - x1) > 1e-6):
        x0 = x1;
        x1 = 0.5 * (x0 + n / x0);
// The square-root of n is approximately x1
```

Implement a MIPS assembly program which implements the above method for single-precision (32-bit) floating point numbers, without using the `sqrt.s` instruction.

(35 marks)