COMP 2604 Assignment 1 (8%) Due: 2nd March 2018 by 12.00pm

Submission Instructions:

- Work in groups of 5
  - form your own groups and submit one assignment per group
  - Ensure each student name and ID is clearly stated on a cover page
- Implement your program for question 1 on a UNIX like platform such as Ubuntu Linux.
- The source code must be appropriately commented and structured to allow users to understand your code
- Upload a zipped file containing your solutions to myelearning before the deadline.
- Absolutely no late submissions

Question 1

The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm:

$$n = \begin{cases} n/2, & \text{if } n \text{ is even} \\ 3 \times n + 1, & \text{if } n \text{ is odd} \end{cases}$$

The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, the sequences for various values of *n* are given below.

n	The sequence
1	1
2	2 1
3	3 10 5 16 8 4 2 1
4	4 2 1
5	5 16 8 4 2 1
6	6 3 10 5 16 8 4 2 1
7	7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
8	8 4 2 1
9	9 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Write a C program, **p1.c**, using the **fork()** system call that generates the sequence of the Collatz conjecture in the child process. The number **n** will be provided in the command line, where **1 ≤ n ≤ 9**. For example, if **n=8** is provided, the numbers **8, 4, 2, 1** in the Collatz sequence will be output by the child process. The child process will output the sequence by calling the uer-defined function **Collatz().** The parent must invoke the **wait()** system call to wait for the child process to complete before exiting the program. Perform necessary error checking to ensure that a positive number is passed on the command line.

Question 2

Consider the following set of five processes P_1, P_2, P_3, P_4, and P_5. The priorities and the lengths of the CPU-burst time in milliseconds are given below.

Process	Burst Time	Priority
P_1	15	5
P_2	3	2
P_3	9	1
P_4	6	4
P_5	4	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.
A larger priority number implies a higher priority.
Draw a Gantt chart illustrating the execution of these processes and compute the average waiting time (AWT) of each of the following CPU scheduling algorithms.

(a) FCFS
(b) Non-preemptive Shortest Job First
(c) Non-preemptive priority scheduling
(d) Round robin with a 5ms time quantum

Question 3:

How many child processes are created if the following program is run?
What is the output that is printed?

```
main (int argc , char ** argv) {
      calc(9)
}
void calc (int n) {
   if (n > 0) {
      fork( );
      printf("%d " , n);
      calc(n-1);
   }
}
```

Question 4:

Consider the following program:

```
main (int argc, char ** argv) {
   int child = fork();
   int x = 100;

   if (child == 0) {
      x -= 10;
   }
   else {
      child = fork();
      x -= 20;
      if (child) {
         x -= 10;
      }
   }
}
```

How many different copies of the variable x are there? What are their values when their process finishes?