JACOBS UNIVERSITY BREMEN
BACHELOR OF COMPUTER SCIENCE

OS 2022 Problem Sheet #3

Student name: *Joshua Law*

Course: *CO-562 Operating Systems* – Professor: *Dr. Jurgen Schonwalder*
Due date: *September 29th, 2022*

Problem 3.1: *readers / writers problem*

Below are three incorrect solutions of the readers-writers problem. Explain in which situations the solutions fail to work correctly. The solutions use the following common definitions:

```
shared object data;
shared int readcount = 0;
semaphore mutex = 1, writer = 1;
```

a)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    up(&mutex);
    if (readcount == 0) up(&writer);
}
```

```
void writer()
{
    down(&writer);
    write_shared_object(&data);
    up(&writer);
}
```

b)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    if (readcount == 0) {
        up(&mutex);
        up(&writer);
    } else {
        up(&mutex);
    }
}
```

```
void writer()
{
    down(&writer);
    write_shared_object(&data);
    up(&writer);
}
```

```
c) void reader()                                  void writer()
   {                                              {
       down(&mutex);                                  down(&writer);
       readcount = readcount + 1;                     down(&mutex);
       if (readcount == 1) down(&writer);             write_shared_object(&data);
       up(&mutex);                                    up(&mutex);
       read_shared_object(&data);                     up(&writer);
       down(&mutex);                              }
       readcount = readcount - 1;
       if (readcount == 0) up(&writer);
       up(&mutex);
   }
```

Answer.

a) In solution a, at the last if statement, up is used on mutex before the up is used on writer, this creates a concurrency issue where if a writer will be called it would fail as the mutex indicates the process is in use.

b) In solution b, up is called on mutex before writer during the last if statement so just like solution a a concurrency issue occurs where the mutex indicates it is in use while writer starts its process, which would present the mutex in the wrong state.

c) In solution c, writer calls down and up on the mutex before and after the writeshare-dobject, which could mess up the mutex because it is unnecessary for the writer to call the mutex, it might cause an unwanted loop in the mutex state.

Problem 3.2: *perfect numbers (multi threading)*

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For example, 6 has the positive divisors `1, 2, 3` and `1 + 2 + 3 = 6`. Write a C program called perfect that finds perfect numbers in a range for numbers. The default number range is `[1, 10000]`. The program accepts the `-s` option to set the lower bound and the `-e` option to set the higher bound. Hence, the invocation perfect `-s 100 -e 1000` will search for perfect numbers in the range `[100, 1000]`. The following function can be used to test whether a given number is a perfect number:

```c
static int
is_perfect(uint64_t num)
{
  uint64_t i, sum;

  if (num < 2) {
    return 0;
  }
  for (i = 2, sum = 1; i*i <= num; i++) {
    if (num % i == 0) {
      sum += (i*i == num) ? i : i + num / i;
    }
  }
  return (sum == num);
}
```

a) Write a program that searches for perfect numbers in a range of numbers. Your program must support the `-s` and `-e` options to define non-default search intervals.

```
./perfect -s 100 -e 10000
496
8128
```

b) Implement an option `-t` that can be used to define how many concurrent threads should be used to execute the search. If the `-t` option is not present, then a single thread is used to carry out the search. For debugging purposes, implement an option `-v` that writes trace information to the standard error. Below is an invocation with two threads and a verbose trace.

```
./perfect -t 2 -v
perfect:  t0 searching [1,5000]
perfect:  t1 searching [5001,10000]
6
28
496
8128
perfect:  t0 finishing
perfect:  t1 finishing
```

c) Determine how the `-t` option impacts the execution time. Pick a search interval that is a reasonable load for your computer hardware and then increase the threading level and determine how the execution time changes. Produce a plot presenting the measurements you have obtained and discuss the results.

Answer.