Concurrent Programming

Exercise Booklet 4: Semaphores

1. Given the following threads:

```
thread {                    thread {
  print('A');                 print('E');
  print('B');                 print('F');
  print('C');                 print('G');
}                           }
```

use semaphores in order to guarantee that:

   a) A is printed before F.

   b) F is printed before C.

2. Given

```
thread {                    thread {
  print('A');                 print('R');
  print('C');                 print('E');
}                             print('S');
                            }
```

use semaphores to guarantee that the only possible output is RACES.

3. Consider the following three threads:

```
thread {              thread {              thread {
  print("R");           print("I");           print("O");
  print("OK");          print("OK");          print("OK");
}                     }                     }
```

Use semaphores to guarantee that the output is R I O OK OK OK (we assume that print is atomic).

4. Consider the following threads that share the variables `y` and `z`.

```
global int y = 0, z = 0;
```

```
thread {              thread {
  int x;                y = 1;
  x = y + z;            z = 2;
}                     }
```

   a) What are the possible final values for `x`?

   b) Is it possible to use semaphores to restrict the set of possible values of `x` to be just two possible values?

5. Given

```
thread                thread                thread
  while (true) {        while (true) {        while (true) {
    print('A');           print('E');           print('H');
    print('B');           print('F');           print('I');
    print('C');           print('G');         }
    print('D');         }
  }
```

Add semaphores to guarantee that:

   a) the number of F $\leq$ the number of A

   b) the number of H $\leq$ the number of E

   c) the number of C $\leq$ the number of G

6. We have three threads A, B, C. We wish operation op_C of C be executed only after A has executed op_A and B has executed op_B. How can we synchronize these processes using semaphores?

7. Consider the following two threads:

```
thread                          thread
  while (true)                    while (true)
    print('A');                     print('B');
```

   a) Use semaphores to guaranteee that at all times the number of A's and B's differs at most in 1.

   b) Modify the solution so that the only possible output is ABABABABAB...

8. The following threads cooperate to calculate the value $N2$ which is the sume of the first N odd numbers. The processes share the variables N and $N2$ which are initialized as follows: $N = 50$ and $N2 = 0$.

```
thread {                        thread
  while (N > 0)                   while (true)
    N = N-1;                        N2 = N2 + 2*N + 1;
  print(N2);
}
```

   a) Provide a solution using semaphores that guarantees that the correct value of $N2$ is printed.