CS2106 Tutorial 6

AY 25/26 Sem 1 — github/omgeta

Q1. C must occur before A, and B is free so we have $B \to C \to A = 6$, $C \to A \to B = 36$, $C \to B \to A = 18$

Q2. Code:

```
int arrived = 0; //shared variable
Semaphore mutex = 1; //binary semaphore to provide mutual exclusion
Semaphore waitQ = 0; //for N-1 process to blocks

Barrier( N ) {
  wait( mutex );
  arrived++;
  signal( mutex );
  if (arrived == N )
    signal( waitQ )

  wait( waitQ );
  signal( waitQ );
}
```

Q3. (a.) Two villagers crossing from opposite directions will lead to deadlock.

(b.) Only a single villager can cross at a time, even from the same diretion.

(c.) Introduce variable `crossing = 0` indicating villages crossing in positive or negative diretion. Code:

```
void enter_bridge_direction1()
{
  bool pass=false;
  while(!pass){
    mutex.wait();
    if(crossing>=0){
      crossing++;
      pass=true;
    }
    mutex.signal();
  }
}
void enter_bridge_direction2()
{
  bool pass=false;
  while(!pass){
    mutex.wait();
    if(crossing<=0){
      crossing--;
      pass=true;
    }
    mutex.signal();
  }
}

void exit_bridge_direction1()
{
  mutex.wait();
  crossing--;
```

```
  mutex.signal();
}
void exit_bridge_direction2()
{
  mutex.wait();
  crossing++;
  mutex.signal();
}
```

(d.) Villagers keep crossing from one direction can indefinitely starve villagers from the other direction.

Q4. (a.) If two processes A, B call `GeneralWait()`, then count will decrement to -2 and the mutex will be released. If two process C, D then call `GeneralSignal()`, they will be able to double signal queue which will cause undefined behaviour.

(b.) Code:

```
GeneralWait() {
  wait(mutex);
  count--;
  if (count < 0) {
    signal(mutex);
    wait(queue);
  } // else removed
  signal(mutex);
}

GeneralSignal() {
  wait(mutex);
  count++;
  if (count <= 0)
    signal(queue);
  else // else added
    signal(mutex);
}
```

Q5. Yes;
Case 1 (R grabs right fork, then left): R can eat so no deadlock
Case 2 (R grabs right fork but cannot grab left): person to left has grabbed both and can eat, so no deadlock
Case 3 (R cannot grab right): person to right has taken it but R is blocked trying to acquire it, so person to left can take their right chopstick and eat, so no deadlock