Parallel Programming Languages and Systems, Assignment 1

s1140740

1 Question 1

At the start of the program `P1` one would execute the loop continuously while `x` is not equal to `y` and `P2` would busy wait until `x` is equal to `y`. When `x` becomes 0, that is when `x = x - 1` is executed and `x = 0` is stored in memory from `P1`, there are four possible combinations of values process `P1` reads from memory before it checks whether `x` is equal to `y` as part of the while loop.

1. **`P1` reads its own value of `x` and `y`**

   Thus, `x = 0` and `y = 0`. Since `0 == 0`, `P1` will break out of the loop. There are two possibilities now:

   (a) `P1` executes `y = y + 1` before `P2` executes `<await (x==y);>`

   In this case, `x = 0` and `y = 1`. Since `P1` is done and `x != y` `P2` will never execute and the program will not terminate.

   (b) `P2` executes `<await (x==y);>` before `P1` executes `y = y + 1`

   Depending on when the value of `y` from `P2` is stored in memory this program will terminate with three different values of `y`. Moreover, there is no dependency on `x` in the rest of `P1`, thus its value will be the same in all three cases. After the loop is exited what remains in `P1` is to read the value of `y`, increment it, and store it in memory. Since only the read and the store are memory operations there are only three possible times when value of `y` from `P2` can be stored:

   i. Before the read

   `P1` is going to read value of `y` from `P2`, increment it by one and store it. Thus the program will terminate with `x = 8` and `y = 3`.

   ii. Between the read and the store

   `P1` is going to read its own value of `y`, that is 0, increment it to 1 while `P2` stores 2 into `y`, and then store 1 into `y`. Thus the program will terminate with `x = 8` and `y = 1`.

   iii. After the store

   `P1` is going to read its own value of `y`, that is 0, increment it to 1, and store 1 into `y`. After this, `P2` stores 2 into `y`. Thus the program will terminate with `x = 8` and `y = 2`.

2. **`P1` reads its own value of `x` but value of `y` from `P2`**

   Thus, `x = 0` and `y = 2`. Since all instructions of `P2` have executed, only `P1` remains active. Further, since `0 != 2`, `P1` will enter the body of the loop and read value of `x` as 8. It will therefore start decrementing `x` until it becomes 2 at which point it will exit the loop and the program will terminate with `x = 2` and `y = 3`.

3. **`P1` reads its own value of `y` but value of `x` from `P2`**

   Thus, `x = 8` and `y = 0`. Since `8 != 2`, `P1` will enter the loop and start decrementing `x`. Depending on when the value of `y` from `P2` is stored in memory this program will behave differently:

   (a) `P2` stores `y = 2` before `x` reaches 2

   `P1` will decrement `x` until it becomes 2 when it will exit the loop, increment the value of `y` to be 3 and the program will terminate with `x = 2` and `y = 3`.

   (b) `P2` stores `y = 2` after `x` reaches 2 but before `P1` breaks out of the while loop

   Hence, `x < y` and thus `P1` will decrement `x` to negative infinity and the program will never terminate.

   (c) `P2` stores `y = 2` after `P1` breaks out of the while loop

   This way `P1` is in the same situation as in case 1 (except `x` is 0 now) and depending on when `P2` stores `y` we get three possible outcomes, each with different value of y:

   i. Before the read
      Program terminates with `x = 0`, `y = 3`.

   ii. Between the read and the store
      Program terminates with `x = 0`, `y = 1`.

   iii. After the store
      Program terminates with `x = 0`, `y = 2`.

4. **`P1` reads values of `x` and `y` from `P2`**

   Thus, `x = 8` and `y = 2`. Since all instructions of `P2` have executed, only `P1` remains active. Further, since `8 != 2`, `P1` will decrement `x` until `x == y`, that is until both `x` and `y` are equal to 2 (`P2` is done and there is no one to modify value of `y`). At that point it will exit the loop. Last instruction of `P1` is `y = y + 1` and hence the program will terminate with `x = 2` and `y = 3`.