Concurrent Programming

Exercise Booklet 1: Traces

Note: For this booklet you must assume that assignment is atomic and that the scheduler is fair. Also, you may use "?" for the value of uninitialized variables.

1. Assume that the **print** command is atomic. Show all possible traces of execution of the following program:

```
thread P: {          thread Q: {
  print("Hi");         print("Hi");
  print("Alice");      print("Bob");
}                    }
```

2. Draw the state diagram for the following programs. What values can x take at the end of the execution?

   (a)
```
global int x = 0;

thread P: {              thread Q: {
  int local = x;           int local = x;
  local = local + 1;       local = local + 1;
  x = local;               x = local;
}                        }
```

   (b)
```
global int x = 0;

thread P:              thread Q:
  x = x + 1;             x = x + 1;
```

3. Given the following program:

```
global int x = 0;
global int y = 0;

thread P:              thread Q:
  y = x + 1;             x = y + 1;
```

   (a) Show an execution trace such that at the end $x = 2$ and $y = 1$.

   (b) Is there a trace s.t. $x = y = 1$. Justify your answer.

4. Given the following program:

```
global int n = 0;

thread P: {            thread Q: {
  int local;             int local;
  repeat (5) {           repeat (5) {
    local = n;             local = n;
    n = local + 1;         n = local + 1;
} }                    } }
```

   - Show an execution trace in which the final value of n is 5.

5. Assume that f has an integer root, i.e., $f(x) = 0$ for some integer x. We now propose different programs for finding this root. We consider a program to be correct if, in the case that f does have a root, both threads terminate. For each program indicate whether it is correct or not, justifying your answer.

   Program A:

```
global boolean found;
```

```
thread P: {                  thread Q: {
  int i = 0;                   int j = 1;
  found = false;               found = false;
  while (!found) {             while (!found) {
    i = i + 1;                   j = j - 1;
    found = (f(i) == 0);         found = (f(j) == 0);
} }                          } }
```

Program B:

```
global boolean found = false;

thread P: {                  thread Q: {
  int i = 0;                   int j = 1;
  while (!found) {             while (!found) {
    i = i + 1;                   j = j - 1;
    found = (f(i) == 0);         found = (f(j) == 0)
} }                          } }
```

Program C:

```
global boolean found = false;

thread P: {            thread Q: {
  int i = 0;             int j = 1;
  while (!found) {       while (!found) {
    i = i + 1;             j = j - 1;
    if (f(i) == 0)         if (f(j) == 0)
      found = true;          found = true;
} }                    } }
```

6. Consider the program:

```
global int n = 0;

thread P: {          thread Q: {
  while (n < 2)        n = n + 1;
    print(n);          n = n + 1;
}                    }
```

   (a) Supply the execution traces that print the following sequences: 012, 002, 02.

   (b) Should 2 necessarily appear in the output?

   (c) How many times can 2 appear in the output?

   (d) How many times can 1 appear in the output?

   (e) How many times can 0 appear in the output?

   (f) What is the length of the shortest sequence that can be exhibited?

7. Consider the program:

```
global int n = 0;

thread P:            thread Q:
  while (n < 1)        while (n >= 0)
    n = n + 1;           n = n - 1;
```

   (a) Provide an execution trace in which the loop in the thread on the left is executed exactly once.

   (b) Provide a trace in which the loop in the thread on the left is executed exactly three times.

   (c) Describe a trace in which the loop in the thread on the left does not terminate.

8. Consider the program:

```
global int n = 0;
global boolean flag = false;
```

```
thread P:                thread Q:
  while (!flag)            while (!flag)
    n = 1 - n;               if (n == 0)
                               flag = true;
```

(a) Provide an execution trace in which the program terminates.

(b) What are the possible values of n when the program terminates.

(c) Can the program not terminate?