CS3101: Quiz 2. **UWO, February 26, 2015.**

Student ID number:
Student Last Name:

Guidelines. The quiz consists of two exercises and is a closed book test. All answers should be written in the *answer boxes*. No justifications for the answers are needed, unless explicitly required. You are expected to do this quiz on your own without assistance from anyone else in the class. If possible, please avoid pencils and use pens with **dark ink**. Thank you.

Exercise 1: multiple choice questions In each case, **zero, one or more answers** may be correct; indicate **all correct answers**. For each of the following pseudo-code (using the same syntax and semantics as `Cilk++` or `CilkPlus`) estimate the work or span using the big-oh notation. Your estimate will depend on the following:

- `n` which is an `int` variable,
- W_A and S_A which are the work and span of the function call `A()`, respectively

Note that `void A (void)` is a `C++` function.

1. `/* Program 1 */`

```
for (int i = 0; i < n ; i++) {
    A();
}
```

What is the work of the above code?

(a) $n \times W_A$
(b) W_A

Answer: the *work* is an estimate of the running time on one processor; thus it is proportional to the total number of instructions. Since the the work of function A is W_A, and function A is executed n times, the work of `Program 1` is (in the order of) $n \times W_A$.

2. `/* Program 2 */`

```
for (int i = 0; i < n ; i++) {
    A();
}
```

What is the span of the above code?

(a) $n \times S_A$
(b) S_A

Answer: the *span* is the minimum running time with infinitely processors, i.e. the critical path length. In the above code, the loop is serial. Hence, the i-th iteration cannot start until the the $(i-1)$-th iteration finishes. So the span is equal to the work and thus is (in the order of) $n \times W_A$.

3. `/* Program 3 */`

```
cilk_for (int i = 0; i < n ; i++) {
    A();
}
```

What is the work of the above code?

(a) $n \times W_A$
(b) $W_A \times \log(n)$

Answer: Although this is a parallel for-loop, the work is the same as its serial elision, i.e. $n \times W_A$. Remember that the work is the running time on `one` processor.

4. `/* Program 4 */`

```
cilk_for (int i = 0; i < n ; i++) {
    A();
}
```

What is the span of the above code?

(a) $S_A \times \log(n)$
(b) $S_A + \log(n)$

Answer: This is a parallel for-loop. Recall that cilk_for uses a 2-way divide-and-conquer implementation, so the height (span) of a cilk_for (not taking iterations into account) is $\log(n)$. So the span of above code is $S_A + \log(n)$.

5. `/* Program 5 */`

```
cilk_for (int i = 0; i < n ; i++) {
    for (int j = 0; j < 4; j++) {
      A();
    }
}
```

What is the work of the above code?

(a) $O(n \times W_A)$
(b) $O(W_A \times \log(n))$

Answer: Similar to `Program 3`, the work is $n \times 4 \times W_A$. So the work is $O(n \times W_A)$ (ignoring the numerical constant factor).

6. `/* Program 6 */`

```
cilk_for (int i = 0; i < n ; i++) {
    for (int j = 0; j < 4; j++) {
      A();
    }
}
```

What is the span of the above code?

(a) $O(n \times S_A)$
(b) $O(S_A + \log(n))$

Answer: Similarly to `Program 5`, the span is $\log(n) + 4 \times S_A$. So the work is $O(S_A + \log(n))$ (ignoring the numerical constant factor).

7. `/* Program 7 */`

```
cilk_for (int i = 0; i < n ; i++) {
    cilk_for (int j = 0; j < 4; j++) {
      A();
    }
}
```

What is the work of the above code?

(a) $O(n \times W_A)$
(b) $O(W_A \times \log(n))$

Answer: Similarly to `Program 3`, the work is $n \times 4 \times W_A$. So the work is $O(n \times W_A)$ (ignoring the numerical constant factor).

8. `/* Program 8 */`

```
cilk_for (int i = 0; i < n ; i++) {
    cilk_for (int j = 0; j < 4; j++) {
```

```
        A();
      }
}
```

What is the span of the above code?

(a) $O(n \times S_A)$
(b) $O(S_A + \log(n))$

Answer: Similarly to `Program 5`, the span is $\log(n) + \log(4) + S_A$. So the work is $O(S_A + \log(n))$ (ignore the constant item).

9. When the CPU needs to read or write a memory location, it checks the cache. If it finds it there, then we have a

   (a) cache miss
   (b) cache hit

   Answer: If the data is already in cache, then we have a cache hit, otherwise, we have a cache miss.

10. If a computation re-uses much of the data it has been accessing, we say that this computation displays

    (a) temporal locality
    (b) spatial locality

    Answer: Because we are re-using the same data, so this is temporal locality.

11. If a computation uses multiple words in a cache line before the line is displaced from the cache, we say that this computation displays

    (a) temporal locality
    (b) spatial locality

    Answer: Because we access the data in the same cache line, so this is spatial locality.

12. Consider the following `cilkplus` code:

```
int x = 0;
cilk_for(int i=0; i<2; ++i) {
x++;
}
```

After executing the above code on a multithreaded machine, the value of x is:

(a) 2
(b) undefined

Answer: Notice that there is a data-race in the parallel for-loop body. Two threads maybe will try to update the same variable x at the same time, so the result is UNDEFINED.