CS4402-9535: Quiz 1. **UWO, February 8, 2017.**

Student ID number:
Student Last Name:

Guidelines. The quiz consists of two exercises. The expected duration is 45 minutes. 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.

Master Theorem Cheat Sheet For $a \geq 1$ and $b > 1$, consider the equation

$$T(n) = a\,T(n/b) + f(n). \tag{1}$$

- for any $\varepsilon > 0$ we have:

$$f(n) \in O(n^{\log_b a - \varepsilon}) \implies T(n) \in \Theta(n^{\log_b a}) \tag{2}$$

- for any $k \geq 0$ we have:

$$f(n) \in \Theta(n^{\log_b a} \log^k n) \implies T(n) \in \Theta(n^{\log_b a} \log^{k+1} n) \tag{3}$$

- for any $\varepsilon > 0$ we have:

$$f(n) \in \Omega(n^{\log_b a + \varepsilon}) \implies T(n) \in \Theta(f(n)) \tag{4}$$

Exercise 1: Analyzing `CilkPlus` functions
For each of the following `CilkPlus` functions, give an estimate of the work $T_1(n)$ and an estimate of the span $T_\infty(n)$. For each of these functions, you will proceed as follows:

1. Write the recurrence relation satisfied by $T_1(n)$
2. Use the Master Theorem to deduce an estimate of $T_1(n)$
3. Write the recurrence relation satisfied by $T_\infty(n)$
4. Use the Master Theorem to deduce an estimate of $T_\infty(n)$

```
void F(int *X, int *Y, int n) {
  if (n==1) {
   Y[0] = X[0];
  } else {
    q = n/2;
    F(X, Y, q);
    F(X+q, Y+q, n-q);
}
}
```

Answer 1

1. $T_1(n) = 2T_1(n/2) + \Theta(1)$ [3 marks]
2. $T_1(n) \in \Theta(n)$ [2 marks]
3. $T_\infty(n) = 2T_\infty(n/2) + \Theta(1)$ [3 marks]
4. $T_\infty(n) \in \Theta(n)$ [2 marks]

```
void F(int *X, int *Y, int n) {
  if (n==1) {
   Y[0] = X[0];
  } else {
    q = n/2;
    cilk_spawn F(X, Y, q);
    F(X+q, Y+q, n-q);
   cilk_sync;
}
}
```

Answer 2

1. $T_1(n) = 2T_1(n/2) + \Theta(1)$ [3 marks]
2. $T_1(n) \in \Theta(n)$ [2 marks]
3. $T_\infty(n) = T_\infty(n/2) + \Theta(1)$ [3 marks]
4. $T_\infty(n) \in \Theta(\log(n))$ [2 marks]

```
void F(int *X, int *Y, int n) {
  if (n==1) {
   Y[0] = X[0];
  } else {
    q = n/2;
    cilk_spawn F(X, Y, q);
    F(X+q, Y+q, n-q);
    cilk_sync;
    for (int k=0; k<n; k=k+1) {
       Y[k]=Y[k]+1
    }
  }
}
```

Answer 3

1. $T_1(n) = 2T_1(n/2) + \Theta(n)$ [3 marks]
2. $T_1(n) \in \Theta(n \log n)$ [2 marks]
3. $T_\infty(n) = T_\infty(n/2) + \Theta(n)$ [3 marks]
4. $T_\infty(n) \in \Theta(n)$ [2 marks]

```
void F(int *X, int *Y, int n) {
  if (n==1) {
   Y[0] = X[0];
  } else {
    q = n/2;
    cilk_spawn F(X, Y, q);
    F(X+q, Y+q, n-q);
    cilk_sync;
    cilk_for (int k=0; k<n; k=k+1) {
       Y[k]=Y[k]+1
    }
  }
}
```

Answer 4

1. $T_1(n) = 2T_1(n/2) + \Theta(n)$ [3 marks]
2. $T_1(n) \in \Theta(n \log n)$ [2 marks]
3. $T_\infty(n) = T_\infty(n/2) + \Theta(\log n)$ [3 marks]
4. $T_\infty(n) \in \Theta(\log^2 n)$ [2 marks]

Julia Cheat Sheet The following `Julia` functions illustrate language constructs for expressing concurrency.

```
@everywhere function fib(n)
                if (n < 2) then
                    return n
                else return fib(n-1) + fib(n-2)
                end
            end

@everywhere function fib_parallel(n)
          if (n < 40) then
              return fib(n)
          else
              x = @spawn fib_parallel(n-1)
              y = fib_parallel(n-2)
              return fetch(x) + y
          end
       end

function producer()
   produce("start")
   for n=1:2
       produce(2n)
   end
   produce("stop")
end

p = Task(producer);

[consume(p) for i=1:4]
```

Exercise 2: writing a parallel `Julia` function

The *Padovan sequence* is the sequence of integers $P(n)$ defined by the initial values and recurrence relation below:

$$P(1) = P(2) = P(3) = 1 \text{ and } P(n) = P(n-2) + P(n-3). \qquad (5)$$

The first few values of $P(n)$ are $1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, 28, 37, 49, 65, 86, 114, 151, 200, 265$.

1. Write a serial `Julia` function, that on an input positive integer `n`, computes the n-th Padovan number $P(n)$. [20 marks]

   Answer 5

```
@everywhere function Padovan_serial(n)
          if (n < 4)
              return 1
          else return Padovan_serial(n-3) + Padovan_serial(n-2)
          end
       end
```

2. Write a parallel `Julia` function, that on an input positive integer `n`, computes the n-th Padovan number $P(n)$. The parallel constructs to be used are `@spawn` and `fetch`. [20 marks]

   Answer 6

```
@everywhere function Padovan_parallel(n)
   if (n < 40)
       return Padovan_serial(n)
   else
       x = @spawn Padovan_parallel(n-3)
       y = Padovan_parallel(n-2)
       return fetch(x) + y
   end
end
```

3. Write a parallel `Julia` task `t`, that generates the sequence of the Padovan numbers. In other words, calling `consume(t)` repeatedly, produces the Padovan numbers successively. [20 marks]

Answer 7

```
function Padovan()
   p1 = 1
   produce(p1)
   p2 = 1
   produce(p2)
   p3 = 1
   produce(p3)
   while true
      p4 = p1 + p2
      produce(p4)
      p1 = p2
      p2 = p3
      p3 = p4
   end
end

s = Task(Padovan)
```