Concurrence, Parallelism and Distributed Systems (CPDS)
Module I: Concurrency
Facultat d'Informàtica de Barcelona
Final Exam
April 24th 2014

Answer the questions concisely and precisely
Answer each problem in a separate page (remember to put your name)
Closed-book exam
Duration: 2 hour

Exercise 1 FSP (3 Points)
Consider the following `FACTORY` assembling three parts `make_A`, `make_B` and `make_C` into a final `output`:

```
MAKER_A=(make_A->ready->restart->MAKER_A).
MAKER_B=(make_B->ready->restart->MAKER_B).
ASSEMBLER_A_B =(ready->assemble_A_B->ready_two->ASSEMBLER_A_B).
ASSEMBLER=(ready_two->make_C->assemble_A_B_C->output->restart->ASSEMBLER).

||FACTORY=(MAKER_A||MAKER_B||ASSEMBLER_A_B||ASSEMBLER)\{ready, ready_two}.
```

Following some questions:

- (1 Point) Give a picture the LST corresponding to `FACTORY`. Comment briefly the result.
- (0.50 Point) Give a picture of the preceding LST after minimising (pressing the button $\mathcal{M}$). Explain intuitively the result.
- (0.50 Punts) Explain the meaning of the operator `<<`. Give the LTS corresponding to :

  ```
  ||PRIORITY_SYNC = FACTORY<<{make_A}.
  ```

- (0.5 Points) Give a safety property `ORDER_A_B_C` asserting that parts A and B are always assembled before than part C is produced. Give a FSP expression to test if `FACTORY` verifies the property
- (0.5 Points) Write a progress property `NON_STOP` to asserting that the production of outputs never stops.

Exercise 2 JAVA (3 Points)
Let us consider a JAVA program `MaxThreeThreads` allowing us to compute the max of and array like `v` $= \{8, 7, 6, 5, 4, 3, 2, 1, 5, 3, 2, 6, 2\}$ for `v.length` $\geq$ 4. The program `MaxThreeThreads` divides `v` in two halves, finding the max of each part. Later, it takes the max of both parts.

Given a sub-array we use the following class to store the current max and to mark the end of the computation (of this max).

```
public class MaxBox {
   private int localMax;
   private boolean localDone;

   public MaxBox(){
      localDone=false;
   }
   public void setMax(int v){
      localMax =v;
   }
   public int getMax(){
      return localMax;
   }
   public synchronized boolean done(){
      localDone=true;
      notifyAll();
      return localDone;
   }
   public boolean getDone(){
      return localDone;
   }
   public synchronized String chekDone()throws InterruptedException {
      while(!localDone)wait();
      return "it is done!!";
   }
}
```

Follows the program `MaxThreeThreads`:

```
public class MaxThreeThreads {

   public static void main(String[] args)throws InterruptedException{
      int[] v= {8, 7, 6, 5, 4, 3, 2, 1, 5, 3, 2, 6, 2};
      int len=v.length;
      MaxBox boxL =new MaxBox();
      MaxBox boxR =new MaxBox();
      ThreadMaxInterval findMaxL = new ThreadMaxInterval(v, 0, len/2, boxL);
      ThreadMaxInterval findMaxR = new ThreadMaxInterval(v, len/2, len, boxR);
      findMaxL.start();
      findMaxR.start();
      ThreadTwoMax findMax =new ThreadTwoMax(boxL, boxR);
      findMax.start();
   }
}
```

Program `MaxThreeThreads` outputs:

```
Max of the left part: 8
Max of the right part: 6
Max of the total: 8
```

(1 Point) Complete the following code corresponding to `ThreadMaxInterval`. This thread finds the max of the subarray v[i..j] (look at the constructor) and store the results in a `MaxBox`. It also updates `localDone`.

```
public class ThreadMaxInterval extends Thread{
   int[] array;
   MaxBox box;
   int k; //lower bound of the interval
   int l; //upper bound of the interval
   ThreadMaxInterval(int[] v, int i, int j, MaxBox b){ ... }
   public void run(){ ... }
}
```

(1 Point) Consider the following the program

```
public class WrongMaxThreads {
   public static void main(String[] args)throws InterruptedException{
      int[] v= {8, 7, 6, 5, 4, 3, 2, 1, 5, 3, 2, 6, 2};
      int len=v.length;
      MaxBox boxL =new MaxBox();
      MaxBox boxR =new MaxBox();
      ThreadMaxInterval findMaxL = new ThreadMaxInterval(v, 0, len/2, boxL);
      ThreadMaxInterval findMaxR = new ThreadMaxInterval(v, len/2, len, boxR);
      findMaxL.start();
      findMaxR.start();
      System.out.println("Max of the left part: " + boxL.getMax());
      System.out.println("Max of the right part: " + boxR.getMax());
      System.out.println("Max of the total: " + Math.max(boxL.getMax(),boxR.getMax()));
   }
}
```

A possible result of the execution is:

```
Max of the left part: 8
Max of the right part: 0
Max of the total: 8
```

Explain shorty why we get this wrong result.

(1 Point) Complete the following thread in a way that MaxThreeThreads works correctly and display the preceding result.

```
public class ThreadTwoMax extends Thread{
   MaxBox L;
   MaxBox R;
   ThreadTwoMax(MaxBox left, MaxBox right){ ... }
   public void run(){ ... }
}
```

Exercise 3 Erlang (3 Points)
The following program find the max of a list

```
my_max([H|T]) -> my_max(T, H).

my_max([H|T], Max) when H > Max -> my_max(T, H);
my_max([_|T], Max)              -> my_max(T, Max);
my_max([],    Max)              -> Max.
```

(2 Points) Desing a pmax(L) such that.

- When L has less than 10 elements it calls my_max, otherwise:
- It halves L into L1, L2.
- It creates two processes P1 and P2. The list L1 goes to P1 and L2 goes to P2. Process P1 uses my_max to find the max and send back this value. Process P2 do the same with L2.

- Suppose that `Max1` and `Max2` store the values received from `P1` and `P2`, use `my_max` to compute and return the max.

(1 Point). Describe shortly the (possible) advantages or disadvantages of `pmax(L)` in relation to `my_max`. In fact which program is faster in your opinion `my_max` or `pmax`?

Exercise 4 (1 Point) Why the modelling of the concurrent processes in LTS is interesting? Explain shortly.