CSE 428, Spring 2000. Midterm 2, 4 April 2000

The exam consists of 5 exercises, totalling 100 points. Write your answers directly on these pages. **Please remember to write your name and SSN at the top of each sheet.**

Exercise 1 (10 pts)

Define in ML a function `factorials : int list -> int list` which, given a list of integer numbers, returns the list obtained by replacing all the positive numbers with their factorials, and the negative numbers with 0. Examples:

```
- factorials [];
val it = [] : int list
- factorials [3,~2,4,0,~5];
val it = [6,0,24,1,0] : int list
```

Exercise 2 (20 pts)

1. (6 pts) Define in ML a datatype `'a btree` whose elements are binary trees, possibly empty. Both the intermediate nodes and the leaves contain information of type `'a`. For instance:

```
          T1                      T2

          1                      ``a''
         / \                     /   \
        /   \                   /     \
       2     3              ``b''    ``c''
      /     / \                        /
     /     /   \                      /
    4     5     6                  ``d''
           \
            \
             7
```

2. (14 pts) Define a function `height : 'a btree -> int` which returns the height of a binary tree, namely the maximum number of nodes in a branch from root to leaf. The height of an empty tree is 0. For example, if `T1` and `T2` are defined as above, we should have:

```
- height T1;
val it = 4 : int
- height T2;
val it = 3 : int
```

Exercise 3 (25 pts)

Consider the following definition of a class Queue in C++

```
class Queue {

private:
   class Element {
   public:
      int info;
      Element* next;
      Element(int k, Element* n) { info = k; next = n; }
   };

   Element* front;
   Element* rear;

public:
   Queue() { front = rear = NULL; }
   ~Queue() { while (front!=NULL) pop(); }
   void insert(int k) {
      if (rear==NULL)
         rear = front = new Element(k,NULL);
      else
         rear = rear->next = new Element(k,NULL);
   }
   int pop() { // the case of an empty Queue is not considered here,
               // but you should consider it in your definition of Queue
      Element* temp = front;
      int n = front->info;
      front = front->next;
      if (front==NULL) rear = NULL;
      delete temp;
      return n;
   }
};
```

Define an analogous class Queue in Java, for multithread use (i.e. such that the same queue can be used by several threads). Make sure that a thread executing a `pop()` on an empty queue is suspended until some other thread inserts an element in the queue.

Exercise 4 (25 pts)

Suppose that the class Queue previously defined is enriched with the following method, which pops and prints all the elements of the queue:

```
public void popAndPrintAll(){
   while (front!=null) System.out.println(pop());
}
```

Consider now the following program in Java:

```
import Queue; // import the definition of the class Queue

class MyThread extends Thread{
   private Queue myQueue;
   public MyThread(Queue q){ myQueue = q; }
   public void run(){
      myQueue.popAndPrintAll();
   }
}

class TestQueue2 {
   public static void main(String[] args){
      int i;

      Queue q1 = new Queue();
      for(i=0;i<1000;i++) q1.insert(i);

      Queue q2 = new Queue();
      for(i=1000;i<2000;i++) q2.insert(i);

      MyThread T1 = new MyThread(q1);
      MyThread T2 = new MyThread(q2);
      T1.start();
      T2.start();
   }
}
```

If you run such a program, you may observe that the printing of the content of `q1` and of `q2` (executed by the threads `T1` and `T2` respectively) is interleaved.

For each of the following proposals, say whether or not it would be a solution to the problem of interleaving, namely whether it would encapsulate the printing sequences so that the printer first prints the content of one queue, and then the content of the other.

1. (5 pts) Use a semaphore `s` shared by the two threads, initialized to 1. In `run()`, perform `s.down()` before the command `myQueue.popAndPrintAll()`, and `s.up()` afterwards.

2. (5 pts) Like before, but with the semaphore initialized to 2.

3. (5 pts) Use a semaphore s initialized to 1. In `main()`, replace the sequence

```
T1.start();
T2.start();
```

with the sequence

```
s.down();
T1.start();
s.up();
s.down;
T2.start();
s.up();
```

4. (5 pts) Add "`synchronized`" to the declaration of the method `run()`.

5. (5 pts) Add "`synchronized`" to the declaration of the method `popAndPrintAll()`.

Exercise 5 (20 pts)

Consider the following Java program:

```
import Queue;     // import the definition of the class Queue

class MyThread extends Thread{
   private Queue q;
   private int value;
   public MyThread(Queue que, int n) { q = que; value = n;}
   public void run() {
      try { // pop() may throw interruptedException, so we need to catch it
         sleep((long)(Math.random()*1000));
         q.insert(value);
         sleep((long)(Math.random()*1000));
         q.insert(value);
         sleep((long)(Math.random()*1000));
         System.out.println(q.pop());
         sleep((long)(Math.random()*1000));
         System.out.println(q.pop());
      }
      catch (InterruptedException e) { }  // if interrupted, do nothing
   }
}

public class TestQueue {
   public static void main(String[] args){
      Queue q = new Queue;
      MyThread T1 = new MyThread(q,1);
      MyThread T2 = new MyThread(q,2);
      T1.start();
      T2.start();
   }
}
```

1. (2 pts) How many numbers will the program print in one execution?

2. (3 pts) Will the numbers be printed in the same order in all the possible executions?

3. (15 pts) If the answer to previous question is "yes": what is the output sequence? If the answer to previous question is "no": what are all the possible output sequences?