CSE 428, Spring 1999. Midterm 2, 7 April 1999

The exam consists of 4 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

Consider the following C++ declaration of the class `stack_or_queue`

```
#include <iostream.h>

template <class A> class stack_or_queue {
   protected:
      class element {
         public:
            A info;
            element* next;
            element(A x, element* p) { info = x; next = p; }
      };
      element* first;
   public:
      stack_or_queue() { first = NULL; }
      bool is_empty() { return first == NULL; }
      A pop() {
         if (is_empty()) {
            cout << "error" << endl;
            return; //do not worry about returning a value in this case
         }
         A inf = first->info;
         element* temp = first;
         first = first->next;
         delete temp;
         return inf;
      }
};
```

1. **(5 pts)** Define a template class `stack`, parametric on `A`, as a derived class of `stack_or_queue`. `stack` should contain an additional method `push(A)` and behave like a LIFO buffer.

2. **(20 pts)** Define a template class `queue`, parametric on `A`, as a derived class of `stack_or_queue`. `queue` should contain an additional method `insert(A)` and behave like a FIFO buffer. The operation of insert should be done in constant time, i.e. without scanning the whole list of elements. You can add other fields and override the method `pop()` if necessary (but it should still execute in constant time as it is in `stack_or_queue`).

3. **(5 pts)** Define a destructor method for the class `stack_or_queue`. You can define also a destructor method in `element` if you find it convenient to use it for the definiton of `~stack_or_queue()`.

Exercise 2

Consider the Java program `Producers_Consumers` seen in class (see attached sheet).

1. **(10 pts)** For each of the following modifications, say whether the program is still correct (i.e. interference-free, and modeling the problem producers-consumers correctly). Justify your answer.

(a) Use `if` instead of `while` in `Buffer.get()`

(b) Modify `Producer.run()`, `Buffer.put()`, and the visibility of `is_empty` in the following way:

```
class Producer extends Thread {
...
  public synchronized void run() {
    try {
      while (!isInterrupted()) {
        sleep((long)(java.lang.Math.random()*1000));
        while (!buff.is_empty) wait();
        buff.put(id);
        notifyAll();
      }
    } catch (InterruptedException e) { return; }
  }
}

class Buffer {
...
  public boolean is_empty;
...
  public void put(int k)
  {
    content = k;
    is_empty = false;
  }
...
}
```

2. **(10 pts)** For each of the following modifications, say whether the program is still deadlock-free. Justify your answer.

   (a) Create one producer and one consumer only (instead of three and two respectively), and use `notify()` instead of `notifyAll()` in both the methods `Buffer.put()` and `Buffer.get()`

(b) Put the statement `notifyAll()` at the beginning of the get() method (i.e. before the statement `while (is_empty) wait()`).

3. **(20 pts)** Modify the definition of the class `Buffer` so to have a two-positions buffer, behaving according to the FIFO discipline.

Exercise 3

1. **(8 pts)** Define in ML a function `count : ''a * ''a list -> int` which, given an element x and a list L, it returns the number of occurrences of x in L. Examples:

```
- count(1,[]);
val it = 0 : int
- count(1,[1,2,3]);
val it = 1 : int
```

```
- count("a",["b","a","c","a"]);
val it = 2 : int
```

2. **(2 pts)** Is it possible to define `count` so to have a more general type, i.e. `count : 'a * 'a list -> int`? Motivate your answer.

Exercise 4

1. **(5 pts)** Define in ML a datatype `'a tree` representing trees with an arbitrary number of subtrees (i.e. such that each node can have any number of children).

2. **(15 pts)** Define a function `height : 'a tree -> int` which returns the length of the longest path from root to leaf (or, equivalently, the number of levels in the tree). You can define auxiliary functions if you want. Examples:

```
     a
    / \
   b   c
  /|\
 d e f_
    /|\ \
   g h i l
       |
       m

height = 5
```

```
   _a_
 / /|\ \
b c d e f
|  / \  |
g h   i l
  |   |
  m   n

height = 4
```

```java
/*              File Producers_Consumers.java                       */
/*   Three Producers and two consumers on a one-position buffer   */

/* ******** Main program ******** */

public class Producers_Consumers {
  public static void main(String[] args) {
    Buffer b = new Buffer();
    new Producer(b,1).start();
    new Producer(b,2).start();
    new Producer(b,3).start();
    new Consumer(b,1).start();
    new Consumer(b,2).start();
  }
}

/* ********** Producer ********* */

class Producer extends Thread {
  private Buffer buff;
  private int id;
  public Producer(Buffer b, int n) { buff = b; id = n; }
  public void run() {
    try {
      while (!isInterrupted()) {
        sleep((long)(java.lang.Math.random()*1000));
        buff.put(id);
      }
    } catch (InterruptedException e) { return; }
  }
}

/* ********** Consumer ********** */

class Consumer extends Thread {
  private Buffer buff;
  private int id;
  public Consumer(Buffer b, int n) { buff = b; id = n; }
  public void run() {
    try {
      while (!isInterrupted()) {
        sleep((long)(java.lang.Math.random()*1000));
        int k = buff.get();
        System.out.println("consumer " + id + " has retrieved the datum put by producer " +
      }
    } catch (InterruptedException e) { return; }
  }
}
```

```
/* *********** Buffer ************ */

class Buffer {
  private int content;
  private boolean is_empty;
  public Buffer() { is_empty = true; }
  public synchronized void put(int k)
    throws InterruptedException
  {
    while (!is_empty) wait();
    content = k;
    is_empty = false;
    notifyAll();
  }
  public synchronized int get()
    throws InterruptedException
  {
    while (is_empty) wait();
    is_empty = true;
    notifyAll();
    return content;
  }
}
```