CSE 428, Section 2, Spring 2002. Final exam, 2 May 2002

This exam consists of 5 questions, totaling 100 points. Write your answers directly on these pages. **Please remember to write your name and student ID at the top of each sheet. If you are in Section 1 (morning section) then cross out the phrase "Section 2" and write "Section 1" instead.** This exam is *closed book*.

Question 1, Java (20 points)

For each of the following sentences about Java, say whether it is true or false.

1. If a class has synchronized methods, then each object of that class will have its own lock.

2. If a thread wants to execute a synchronized method on a certain object, it will first have to acquire the locks of all the objects of that class.

3. A command of the form `t.wait()` will suspend the thread which is executing the command (assuming that the method `wait()` is the standard one, i.e. it has not been redefined).

4. A suspended thread can only be resumed by a thread **of the same class** executing a `notify()` or `notifyAll()`.

5. If a thread which is holding a lock executes a `wait()` then there is a deadlock.

6. Weak fairness means that if an action (or thread) is always enabled, it will be executed sooner or later.

7. In event programming, the program in general needs to specify when the events should occur, and in which order.

8. In event programming, a typical event would be the motion of the mouse.

9. If an exception is "checked", then it can never be raised at compile time.

10. An exception like "NumberFormatException", when is raised, will automatically send the control back to the instruction that has raised the exception, and such instruction will be executed again.

Question 2, Memory management (20 points)

Consider the following C++ program, which contains the definition of a class queue:

```
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; }
   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;
   }
};
void insertnumbers(Queue* q, int n){
   if (n==0)
      cout<<''Checkpoint 1\n'';
   else {
      q->insert(n);
      insertnumbers(q, n-1);
   }
};
void aux() {
   Queue q();
   insertnumbers(&q,3);
   cout << ``Checkpoint 2\n'';
}

void main() {
   aux();
   cout << ``Checkpoint 3\n'';
}
```

Illustrate with three drawings the situations in memory (stack and heap) at the moment in which the instructions `cout << "Checkpoint 1 \n";`, `cout << "Checkpoint 2 \n";`, and `cout << "Checkpoint 3 \n";` are executed, respectively. Please make explicit all the informations stored in the activation records and in the (allocated part of the) heap at that moment. Use arrows for pointers. Remember that in C++ there is no static link. You can also omit the control link and the temporaries.

Please state explicitly whether there are memory leaks or dangling pointers.

Question 3, Basic ML programming (20 points)

Part A: Consider the following datatype for mobiles that contain integer valued weights at their leafs.

```
datatype mobile = Weight of int | Node of mobile * mobile;
```

Write in ML a definition of the function `balanced : mobile -> bool`, which returns true if and only if its argument is a balanced mobile; that is, it is a mobile such that for every node, the total weight of its left subtree is equal to the total weight of its right subtree. For example,

```
balanced(Node (Node(Weight 2, Weight 2), Node(Weight 2, Weight 2)));
```

returns true and

```
balanced(Node (Node(Weight 2, Weight 2), Node(Weight 1, Weight 2)));
```

returns false.

Part B: Consider the following datatype for simple arithmetic expressions.

```
datatype exp = I of int | Plus of exp * exp | Times of exp * exp
                        | Square of exp;
```

Write in ML a definition of the function `evaluate : exp -> int`, which returns returns the value of an expression: here, `Plus` denotes addition, `Times` denotes multiplication, and `Square` denotes the squaring operation. For example,

```
evaluate (Square (Plus (Times (I 1, I 10), Times (I 3, I 2))));
```

would return the value $((1 \times 10) + (3 \times 2))^2$ namely 256.

Question 4, Types and higher-order programming in ML (20 points)

Part A: Write in SML code to define the function `iterate` which takes a function and an integer n and a value and applies that function to the value n-times. In other words, the value of

`iterate` $f\ n\ x$

should equal $f^n(x)$. If $n = 0$, then the result of composing should be the identify function; that is, (`iterate` f 0 x) should equal x. Also, for example, (`iterate` f 3 x) should equal $f(f(f(x)))$. (You may assume that $n \geq 0$.) displayed value should be simply x. **Also:** after writing down the function's definition, indicate what type ML will infer for this function. (Notice that the function is written in Curried syntax.)

Part B: The following code declares one datatype and three functions.

```
datatype numb = R of real | C of real * real;

fun cases f g (R x) = f x
  | cases f g (C(x,y)) = g(x,y);

fun project (R x) = x
  | project (C(x,_)) = x;

fun plus(R x,   R y   ) = R(x+y)
  | plus(R x,   C(u,v)) = C(x + u, v)
  | plus(C(x,y),C(u,v)) = C(x+u, y+v)
```

Write down the type for the three defined functions that is inferred by ML. If there is a type error, then indicate that. If there are any warning printed, then indicate that as well.

1. The type for `cases`: (list also any error or warning messages)

2. The type for `project`: (list also any error or warning messages)

3. The type for `plus`: (list also any error or warning messages)

Question 5, Basic Prolog programming (20 points)

Write code for this question in Prolog. If you need any auxiliary predicates that are not built-in, then you will need to define those predicates. (Example of builtins that you may use are the comparison predicates such as `<` and `>` and `=` and `>=` and the `is` predicate).

1. Write code to specify the predicate `pick(List,YesNo,Selected)` so that this holds if and only if `List` is a list, `YesNo` is a list the same length as `List` which contains only the constants `yes` and `no`, and `Selected` is a list of the members in `List` for which the correspond member of `YesNo` is `yes`. For example, we would have the following query and result.

```
?- pick([5,8,34,55,3,9],[yes,no,yes,yes,no,no],L).
L = [5, 34, 55] ;
No
?-
```

2. Write code to specify the predicate `union(ListOfLists,List)` so that this holds if and only if `ListOfLists` is a list of lists and `List` is a list that results from appending all the members of `ListOfLists` together in the order that they appear in that list. (If you use the `append` predicate, you do not need to define it.) For example, we would have the following query and result.

```
?-  union([[3,4],[6],[],[[67],3]],L).
L = [3, 4, 6, [67], 3] ;
No
?-
```

3. Write code to specify the predicate `minlist(List,Min)` so that this holds if and only if `List` is a list of integers between 0 and 1000 inclusively and `Min` is the smallest member of `List`. You may assume that the min of the empty list can be taken as 1001. For example, we would have the following query and result.

```
?- minlist([45,66,13,989],M).
M = 13 ;
No
?-
```