1. Environment Model [27 pts] (parts a–d)

Consider the following code:

```ocaml
let val y = (ref "hello", "goodbye")
val z = ((#1 y) := (#2 y); 0)
fun f(x: int) = #2 y
fun g(y: int) = ref f
in
  g(5)
end
```

(a) [2 pts] What is the type of \( f \)?

(b) [2 pts] What is the type of \( g \)?

(c) [18 pts] Draw the result produced by evaluating this expression in the environment model.

(d) [5 pts] What garbage (other than environment entries) is generated by evaluating this program?

2. Data abstraction [33 pts] (parts a–d)

Suppose we want to implement a game of N-by-N tic-tac-toe using a mutable data abstraction for the board. The following is a start at an interface:

```ocaml
(* A board is a mutable N-by-N tic-tac-toe board. *)
type board
datatype contents = X | O | Empty
(* A cell is a cell coordinate, from (1,1) up to (N,N) *)
type cell = int * int
(* create(n) creates an n-by-n board with all cells empty. *)
val create: int -> board
(* The number of cells in one row or column of the board. *)
val boardSize: board -> int
(* The number of non-empty cells. *)
val moves: board -> int
(* The contents of a board cell. *)
val getCell: board*cell -> contents
(* Set the contents of a board cell. *)
val setCell: board*cell*contents -> unit
(* Return whose move it is (always X or O) *)
val whoseMove: board -> contents
```

(a) [5 pts] Classify each of these operations as a creator, observer, or mutator.

(b) [7 pts] Supply any missing preconditions.

Consider the following representation:

```ocaml
type board = { size: int,
              X's: cell list ref,
              O's: cell list ref }
```

Using this rep, here is how we might implement the function create so that it takes only \( O(1) \) time in the board size:
fun create(n: int) = { size = n, X's = ref nil, O's = ref nil }

However, some of the other operations are not so easy to implement.

(c) [15 pts] Give an appropriate representation invariant for this representation. Think about what will be needed to implement all of the functions in the interface above.

(d) [6 pts] Suggest a different representation that would permit all of the operations except create to be implemented in time \(O(1)\) in the board size.

\[
\text{type board = }
\]

3. Recurrences [20 pts] (parts a–b)

The conventional algorithm for multiplying two square matrices of size \(n\) takes \(O(n^3)\) time. However, there is an asymptotically more efficient algorithm in which the matrix is divided into smaller matrices of size \(\frac{n}{2}\) by \(\frac{n}{2}\) and 7 matrix multiplications are performed on them. Thus, we arrive at the following recurrence:

\[
\begin{align*}
T(1) & = 1 \\
T(n) & = 7T(n/2)
\end{align*}
\]

To simplify analysis, let us consider values of \(n\) that are powers of two.

(a) [6 pts] Is the solution to this recurrence \(O(n^3)\)? Justify your answer briefly.

(b) [14 pts] Find the value of \(c\) such that the solution to the recurrence is \(\Theta(n^c)\).

4. Type checking [20 pts] (parts a–c)

(a) [7 pts] Define a function \(f\) with type \((\alpha*\beta)\ ref \Rightarrow (\alpha \ ref)\ast(\beta \ ref)\). Remember that this function must be polymorphic.

\[
\text{fun f(x: (\alpha*\beta) ref) = }
\]

(b) [3 pts] Give an example of two type expressions that contain unsolved type variables but that cannot be unified.

(c) [10 pts] Consider the following SML function:

\[
\text{fun f(x,y,z,w) = if z(x) then (x, (y,z)) else (z(x), w)}
\]

If we let the SML type inference algorithm reconstruct types for this definition, what will be the types inferred for the identifiers \(x, y, z\) and \(w\)?