Name: _____

1. (1 pt.)
   - **Read all material carefully.**
   - *If in doubt whether something is allowed, ask, don't assume.*
   - You may refer to your **books, papers, and notes** during this test.
   - **E-books** may be used *subject to the restrictions* noted in class.
   - **Computers** (including smart phones, tablets, etc.) **are not permitted**, except when used strictly as e-books or for viewing ones own notes.
   - **Network access** of any kind (cell, voice, text, data, ...) is **not permitted**.
   - Write, and draw, carefully. **Ambiguous or cryptic answers receive zero credit.**
   - Use **class and textbook conventions** for notation, algorithmic options, etc.
   - **Do not attach or remove any pages.**

Write your name in the space provided above.
Do not write anything else on this page.

WAIT UNTIL INSTRUCTED TO CONTINUE TO REMAINING QUESTIONS.

(Do not view any other pages.)

Do not write on this page.
(It is for use in grading only.)

Q	Full	Score
1	1	
2	4	
3	3	
4	3	
5	3	
6	3	
7	3	
8	25	
total	45	

2. (4 pts.) This question, and the next few, ask for the result of evaluating the given expression using the `sml` REPL (read-eval-print loop of *Standard ML of New Jersey*). Assume that the expressions are evaluated in the order listed.

   In your response, *draw a* ***box*** *around the* ***type*** *and* `oval` *around the* ***value****.* If there is an error then clearly state the error. **Explain** your answers briefly for better partial credit.

   Reminder: Use of computers and similar devices is not permitted.

```
7 * 36 div 6 div 2 * 3;
```

3. (3 pts.) (See Question 2.)

```
fun f101 (f, nil) = nil
  | f101 (f, h::t)  = f(h) :: f101(f, t);
```

4. (3 pts.) (See Question 2.)

```
f101 ("hello");
```

5. (3 pts.) (See Question 2.)

```
map (fn i => i * i) [3, 1, 4, 1, 5];
```

6. (3 pts.) (See Question 2.)

```
map (fn i => (fn j => i * i + j));
```

7. (3 pts.) (See Question 2.)

```
map map;
```

8. (25 pts.) (Reminder: Read carefully!) Provide a complete JCoCo assembly language program that reads a single line of an arbitrary number of whitespace-separated integers on *standard input* and prints a single line of space-separated integers to *standard output*, where each integer in the output is the square of the corresponding integer in the input. **Explain why your program is correct** using comments and separate text.

[additional space for earlier material]

[additional space for earlier material]