Instructions

- Do not start this test until you are told to do so!
- You have 20 minutes for this quiz.
- This is a closed book quiz. No notes or other aids are allowed.
- For partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.
- This quiz is for 21 points.

1. (7 points) Higher-order OCaml

```ocaml
let rec map f l = match l with
    [] -> []
  | h::t -> (f h) :: (map f t);

let rec fold f a l = match l with
    [] -> a
  | h::t -> fold f (f a h) t;

let add x y = x + y;;
```

Using only the methods provided above, construct an OCaml function called ‘foo’ that takes an ‘int list list’ and returns an ‘int list’ where each element of the output is the sum over the corresponding inner list from the input. Do not use matching or recursion in your implementation.

For example:
```
foo [[6;1;8];[3;5;7;3;2];[8;17];[9;10;11];[]] => [15;20;25;30;0]
```

```ocaml
let foo l = map (fold add 0) l;;
```
2. (7 points) Following the rules below, fill in the missing parts of the derivation that proves the following:

\[; \text{let } y = [1;2] \text{ in } (\text{fun } x = \text{List.hd } x) \ y \Rightarrow 1\]

<table>
<thead>
<tr>
<th>Identifier [Id]</th>
<th>Let binding [Let]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A(x) = v )</td>
<td>( A; E_1 \Rightarrow v_1 )</td>
</tr>
<tr>
<td>( A; x \Rightarrow v )</td>
<td>( A, x = v_1; E_2 \Rightarrow v_2 )</td>
</tr>
<tr>
<td></td>
<td>( A; \text{let } x = E_1 \text{ in } E_2 \Rightarrow v_2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function definition [Fun]</th>
<th>Function application [App]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun } x = E \Rightarrow (\lambda x. E; A) )</td>
<td>( A; E_1 \Rightarrow (\lambda x. E, A') )</td>
</tr>
<tr>
<td></td>
<td>( A; E_2 \Rightarrow v_2 )</td>
</tr>
<tr>
<td></td>
<td>( A', x = v_2; E \Rightarrow v )</td>
</tr>
<tr>
<td></td>
<td>( A; E_1 E_2 \Rightarrow v )</td>
</tr>
</tbody>
</table>

The TA will put a color version on the projector.
3. (7 points) Write the output of the following C code for the different types of parameter passing:

```c
int i = 1;

void foo(int f, int g) {
    f = f + i;
    g = g + f;
}

int main() {
    int a[] = {2, 0, 1};
    foo(i, a[i]);
    printf("%d %d %d %d\n", i, a[0], a[1], a[2]);
}
```

(a) Call by Value

```
1 2 0 1
```

(b) Call by Name

```
2 2 0 3
```

(c) Call by Reference

```
2 2 2 1
```