Name:	
ID:	

CSE 130, Winter 2013: Midterm Examination Feb 12th, 2012

- Do **not** start the exam until you are told to.
- This is a open-book, open-notes exam, but with no computational devices allowed (such as calculators/cellphones/laptops).
- Do **not** look at anyone else's exam. Do **not** talk to anyone but an exam proctor during the exam.
- Write your answers in the space provided.
- Wherever it gives a line limit for your answer, write no more than the specified number of lines. The rest will be ignored.
- Work out your solution in blank space or scratch paper, and only put your answer in the answer blank given.
- The points for each problem are a rough indicator of the difficulty of the problem.
- Good luck!

1.	25 Points	
2.	20 Points	
3.	10 Points	
TOTAL	55 Points	

1. [25 points]

a. [18 points] Consider the following datatype:

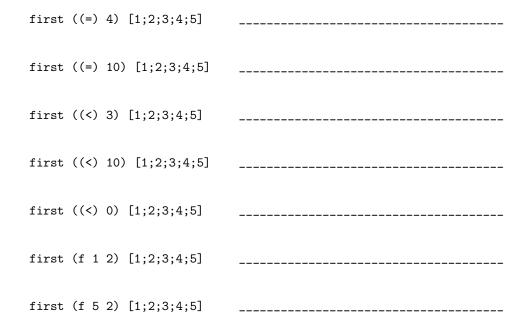
```
type 'a maybe =
    | None
    | Some of 'a
```

In this problem you are going to write first: ('a -> bool) -> 'a list -> 'a maybe. Given a "tester" function f and a list 1, (first f 1) returns the first element of the list 1 for which f returns true. If there is no element in the list for which f returns true, then first returns None. If there are multiple elements in the list for which f returns true, then the first such element is returned. For example:

in List.fold_left fold_fn base 1

b. [**7 points**] Now, consider the following code:

For each expression below, write down what it evaluates to:



2. [20 points]

a. [10 points] You are going to write a function zip: 'a list -> 'b list -> ('a * 'b) list. Given two lists 11 and 12, (zip 11 12) returns a list containing pairs of corresponding elements from 11 and 12. If lists 11 and 12 have different lengths, the returned list has the same length as the shorted of the two lists. For example:

b. [5 points] Recall the map function, which has type ('a -> 'b) -> 'a list -> 'b list. You will now write a function map2: ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list. Given a function f and two lists 11 and 12, (map2 f 11 12) returns a list in which each element is produced by calling f on the corresponding elements of 11 and 12. For example:

```
# (+);;
- : int -> int -> int = <fun>
# map2 (+) [1;2;3] [4;6;8];;
- : int list = [5; 8; 11]
# map2 (-) [1;2;3] [4;6;8];;
- : int list = [-3; -4; -5]
# map2 (/) [10;9;4] [2;3;4];;
- : int list = [5; 3; 1]
# map2 (+) [1;2] [4;6;8];;
- : int list = [5; 8]
```

Using map and zip, write the code for map2 below. Note that map2 is not declared as rec so it cannot call itself. Be careful with currying to make sure that everything typechecks properly.

let map2 f 11 12 =

c. [**5 points**] You will now write a function:

```
map3:('a \rightarrow 'b \rightarrow 'c \rightarrow 'd) \rightarrow 'a list \rightarrow 'b list \rightarrow 'c list \rightarrow 'd list
```

This function works similarly to map 2 but with three lists instead of two. For example:

```
# let add a b c = a + b + c;;
val add : int -> int -> int -> int = <fun>
# map3 add [1;2] [3;4] [5;6];;
- : int list = [9; 12]
# map3 add [1] [3;4] [5;6];;
- : int list = [9]
```

Using map and zip, write the code for map3 below. Note again that map3 is not declared as rec so it cannot call itself. Be careful with currying to make sure that everything typechecks properly.

let map3 f 11 12 13 =

3. [10 points]

You will now write a function unzip: ('a * 'b) list -> 'a list * 'b list. Given a list 1 of pairs, (unzip 1) returns two lists 11 and 12 where 11 contains the first element of each pair in 1 and 12 contains the second element of each pair in 1. For example: