CSE 130, Winter 2012: Midterm Examination
Feb 14th, 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!

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>45 Points</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>15 Points</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>25 Points</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>0 Points</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>85 Points</td>
<td></td>
</tr>
</tbody>
</table>
1. [ 45 points ] We are going to implement merge sort, in small steps.
   a. [ 15 points ] First, you will write a function \texttt{split : 'a list \rightarrow 'a list * 'a list}. This function splits a given list in two parts along the middle. Here are examples of it running:

   \begin{verbatim}
   # split [23;1;8;3];;
   - : int list * int list = ([23; 1], [8; 3])
   # split [23;1;8;3;6];;
   - : int list * int list = ([23; 1], [8; 3; 6])
   # split [23;1;8;3;6;20];;
   - : int list * int list = ([23; 1; 8], [3; 6; 20])
   # split ["a";"b";"c"];;
   - : string list * string list = (["a"], ["b"; "c"])
   # split ["a"];;
   - : string list * string list = ([], ["a"])\end{verbatim}

   To do this, you will use \texttt{fold_left}, whose type is \texttt{(\textquoteleft a -> \textquoteleft b -> \textquoteleft a) \rightarrow \textquoteleft a \rightarrow \textquoteleft b list \rightarrow \textquoteleft a}. You can also assume a function \texttt{length: 'a list \rightarrow int} which returns the length of a list. Now, fill in the implementation of \texttt{split} below:

   \begin{verbatim}
   let rec split l =
       (* additional let declarations if you need any *)

       let base = _______________________________________ in
       let fold_fn (i,l1,l2) elmt =
           ______________________________________

       let (_, l1, l2) = List.fold_left fold_fn base l in
       (l1, l2)
   \end{verbatim}
b. [15 points] You will now write a `merge` function of type `'a list -> 'a list -> 'a list`. This function takes two lists that are already sorted using the ordering `<=`, and merges them into a sorted list. Here are examples of `merge`:

```
# merge [2;4;6;8] [1;3;5];;
- : int list = [1; 2; 3; 4; 5; 6; 8]

# merge [2;10;20] [1;2;3;4;5;8;10;12];;
- : int list = [1; 2; 3; 4; 5; 6; 8; 10; 10; 12; 20]
```

Fill in the implementation of `merge` below:

```
let rec merge l1 l2 =
  match (l1, l2) with
  | ([], l) -> __________________________________________________________________
  | (l, []) -> __________________________________________________________________
  | _____________________________________________________________________________
    _____________________________________________________________________________
    _____________________________________________________________________________
    _____________________________________________________________________________

match (l1, l2) with
  | ([], l) -> __________________________________________________________________
  | (l, []) -> __________________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
  | ____________________________________________________________
```
c. [15 points] We are now ready to write `merge_sort`, whose type is `'a list -> 'a list`. You should use the `split` and `merge` functions above to implement `merge_sort`. Recall that merge sort works by splitting the input list in two, recursively sorting the two sub-lists, and then merging the two results of the recursive calls into a sorted list. Here are examples of running `merge_sort`:

```plaintext
# merge_sort [2;10;3;2;1];;
- : int list = [1; 2; 2; 3; 10]

# merge_sort [-10;0;10;-20;100;-100];;
- : int list = [-100; -20; -10; 0; 10; 100]
```

Fill in the implementation of `merge_sort` below:
```plaintext
let rec merge_sort l =

---------------------------------------------------------------

---------------------------------------------------------------

---------------------------------------------------------------

---------------------------------------------------------------

---------------------------------------------------------------

---------------------------------------------------------------
```
2. [15 points] Assume you are given the following two functions:

\[
\begin{align*}
\text{explode} : \text{string} & \rightarrow \text{char list} \\
\text{implode} : \text{char list} & \rightarrow \text{string}
\end{align*}
\]

Given a string \( s \), \( \text{explode} \ s \) returns the list of characters in the string, and given a list of characters \( l \), \( \text{implode} \ l \) returns a string that contains all the characters in the list. For example:

\[
\begin{align*}
# \text{explode} "abc";; \\
- : \text{char list} = [\'a\'; \'b\'; \'c\']
\end{align*}
\]

\[
\begin{align*}
# \text{implode} [\'a\'; \'b\'; \'c\'];; \\
- : \text{char list} = "abc"
\end{align*}
\]

Also, assume that you have the traditional map, with type \((\text{'a} \rightarrow \text{'b}) \rightarrow \text{'a list} \rightarrow \text{'b list}\). Using map, explode and implode, write a function replace: string \( \rightarrow \) string that replaces the hyphen character, \'-\', with space, \'\ '.


3. [ 25 points ]

a. [ 13 points ] In this problem you are going to write a function:

\[ \text{app} : (\forall a \rightarrow b) \text{ list } \rightarrow a \rightarrow \text{ list} \]

Given a list of functions \( l \), \( \text{app} \ 1 \ x \) returns a list where each element of the list is the application of the corresponding function from \( l \) to \( x \). For example:

\[
\text{let incr x = x+1;};
\text{val incr : int -> int = <fun>}
\]
\[
\text{let decr x = x-1;};
\text{val decr : int -> int = <fun>}
\]
\[
\text{app [incr;decr] 10;}
- : int list = [11; 9]
\]

Implement \text{app} below using map:

\[
\]
\[
\]
\[
\]
\[
\]

b. [ 12 points ] Now, consider the following code:

\[
\text{let [f1;f2] = app [(=);(<)] 2;;}
\text{val f1 : int -> bool = <fun>}
\]
\[
\text{val f2 : int -> bool = <fun>}
\]

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

\[
(f1 \ 1) \ ____________________________
\]
\[
(f1 \ 2) \ ____________________________
\]
\[
(f1 \ 3) \ ____________________________
\]
\[
(f2 \ 1) \ ____________________________
\]
\[
(f2 \ 2) \ ____________________________
\]
\[
(f2 \ 3) \ ____________________________
\]
4. [ 0 points ] Circle the correct answer. When Sorin was a kid, he did which of the following sports:

1. Snow Rugby
2. Lugging
3. Curling
4. Figure Skating
5. Speed Skating