Section-Wide Items

Video Lectures

Homework 1

✓ **Programming Assignment:** Homework 1 (Auto-Grader)
3h

📖 **Reading:** Homework 1 Detailed Guidelines for Peer Assessment

👤 **Peer-graded Assignment:** Homework 1
45 min

💬 **Review Your Peers:** Homework 1

Community-Contributed Resources

Homework 1 Detailed Guidelines for Peer Assessment

Problem 1

Here is a sample solution:

```
1   fun is_older (date1 : int * int * int, date2 : int * int * int) =
2       let
3           val y1 = #1 date1
4           val m1 = #2 date1
5           val d1 = #3 date1
6           val y2 = #1 date2
7           val m2 = #2 date2
8           val d2 = #3 date2
9       in
10          y1 < y2 orelse (y1=y2 andalso m1 < m2)
11                  orelse (y1=y2 andalso m1=m2 andalso d1 < d2)
12      end
```

- Be lenient on how let-expressions are used. It is okay if there are no local val bindings. It is also okay if there are more (e.g., to avoid repeating the expression `y1=y2`).

- For the logic expression, it is okay to use `if ... then ... else ...` instead of `orelse` and `andalso`, but the logic should still be clear: starting by comparing the year, then the month, then the day. If the logic is hard to follow, give a 4 or 3.

Give a 3 for this sort of more imperative looking code:

```
1   fun is_older (date1 : int * int * int, date2 : int * int * int) =
2       let val y1 = #1 date1
3           val m1 = #2 date1
4           val d1 = #3 date1
5           val y2 = #1 date2
6           val m2 = #2 date2
7           val d2 = #3 date2
8       in
9           let val b1 = y1 < y2
10          in
11              if b1
12              then true
13              else let val b2 = y1 > y2
14                   in
15                       if b2
16                       then false
17                       else ...
18                   ...
19      end
20
```

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 2

Here is a sample solution:

```
1   fun number_in_month (dates : (int * int * int) list, month : int) =
2       if null dates
3       then 0
4       else if #2 (hd dates) = month
5       then 1 + number_in_month(tl dates, month)
6       else number_in_month(tl dates, month)
7
```

Make sure the solution has clear recursive calls and clearly evaluates to `0` if `dates` is `null`. The solution does not have to be exactly like the sample above. For example, this solution also deserves a 5:

```
1   fun number_in_month (dates : (int * int * int) list, month : int) =
2       if null dates
3       then 0
4       else (if #2 (hd dates) = month then 1 else 0)
5               + number_in_month(tl dates, month)
6
```

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 3

Here is a sample solution:

```
1   fun number_in_months(dates : (int * int * int) list, months : int list) =
2       if null months
3       then 0
4       else number_in_month(dates, hd months) + number_in_months(dates, tl months)
5
```

- Give a 3 if the solution does not use **number_in_month** as a helper function or if it is substantially longer than a single if-then-else expression.

- Give a 4 if it uses a let expression for not much reason (for a short expression that is used only once). Do this for all the remaining problems (we won't repeat this instruction for each problem).

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 4

Here is a sample solution:

```
1   fun dates_in_month (dates : (int * int * int) list, month : int) =
2       if null dates
3       then []
4       else if #2 (hd dates) = month
5       then (hd dates)::dates_in_month(tl dates, month)
6       else dates_in_month(tl dates, month)
7
```

Give at most a 4 for any solution that uses ML's append operator (the @ character). Otherwise follow similar instructions as for earlier problems.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 5

Here is a sample solution:

```
1   fun dates_in_months(dates : (int * int * int) list, months : int list) =
2       if null months
3       then []
4       else dates_in_month(dates, hd months) @ dates_in_months(dates, tl months)
5
```

Give a 3 if the solution does not use **date_in_month** as a helper function or if it is substantially longer than a single if-then-else expression.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 6

Here is a sample solution:

```
1   fun get_nth (lst : string list, n : int) =
2       if n=1
3       then hd lst
4       else get_nth(tl lst, n-1)
```

Give at most a 3 if the solution uses an algorithm much more complicated than the code above.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 7

Here is a sample solution:

```
1   fun date_to_string (date : int * int * int) =
2       let
3           val names = ["January", "February", "March", "April", "May", "June",
4                        "July", "August", "September", "October", "November",
                         "December"]
5       in
6           get_nth(names,#2 date) ^ " " ^ Int.toString(#3 date)
7           ^ ", " ^ Int.toString(#1 date)
8       end
```

Give at most a 2 if the solution does not use a list of month names in some way. However, you can give a 5 for a solution that puts the list of month names outside the function. Give at most a 4 if the solution does not use **get_nth** with the list of month names as an argument.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 8

Here is a sample solution:

```
1   fun number_before_reaching_sum (sum : int, lst : int list) =
2       if sum <= hd lst
3       then 0
4       else 1 + number_before_reaching_sum(sum - hd lst, tl lst)
```

Any nicely formatted solution of roughly this length is probably good style, but look for the logic of a recursive call with argument `sum - hd lst`, giving at most a 4 if it is difficult to find.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 9

Here is a sample solution:

```
1   fun what_month (day_of_year : int) =
2       let
3           val month_lengths = [31,28,31,30,31,30,31,31,30,31,30,31]
4       in
5           1 + number_before_reaching_sum(day_of_year, month_lengths)
6       end
```

Give at most a 2 if the solution does not use a list of month lengths and the `number_before_reaching_sum` function in some way. However, you can give a 5 for a solution that puts the list of month lengths outside the function.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 10

Here is a sample solution:

```
1   fun month_range (day1 : int, day2 : int) =
2       if day1 > day2
3       then []
4       else what_month day1 :: month_range(day1 + 1, day2)
```

Give at most a 3 for a solution that uses ML's append operator (the @ character). Give at most a 4 for a solution that has more than a single if-then-else expression.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problem 11

Here are two sample solutions:

```
1   fun oldest (dates : (int * int * int) list) =
2       if null dates
3       then NONE
4       else let
5               val ans = oldest(tl dates)
6       in
7           if isSome ans andalso is_older(valOf ans, hd dates)
8           then ans
9           else SOME (hd dates)
10      end
11
12  (* arguably better alternate solution avoiding isSome / valOf *)
13  fun oldest (dates : (int * int * int) list) =
14      if null dates
15      then NONE
16      else let
17              fun f dates =
18                  if null (tl dates)
19                  then hd dates
20                  else let
21                          val ans = f (tl dates)
22                  in
23                      if is_older(ans, hd dates)
24                      then ans
25                      else hd dates
26                  end
27      in
28          SOME(f dates)
29      end
```

Give at most a 3 if oldest could be called recursively twice with the same list (probably `tl dates`). Give at most a 4 if `is_older` is not used.

Remember that you are grading on general style, not how close to the sample solution a student solution is. It is perfectly fine for a solution to be significantly different from the sample, as long as it has good style.

Problems 12 and 13

You do not need to provide feedback on problems 12 and 13 (the challenge problems), but you are welcome to give text feedback on these problems if you wish. Here are sample solutions for the challenge problems although there are other equally good if not better approaches you could take:

```
1   (* quadratic algorithm rather than sorting which is nlog n *)
2   fun mem(x : int, xs : int list) =
3       not (null xs) andalso (x = hd xs orelse mem(x, tl xs))
4   fun remove_duplicates(xs : int list) =
```

```sml
   4   fun remove_duplicates(xs : int list) =
   5       if null xs
   6       then []
   7       else
   8           let
   9               val tl_ans = remove_duplicates (tl xs)
  10           in
  11               if mem(hd xs, tl_ans)
  12               then tl_ans
  13               else (hd xs)::tl_ans
  14           end
  15
  16   fun number_in_months_challenge(dates : (int * int * int) list, months : int
       list) =
  17       number_in_months(dates, remove_duplicates months)
  18
  19   fun dates_in_months_challenge (dates : (int * int * int) list, months : int
       list) =
  20       dates_in_months(dates, remove_duplicates months)
  21
  22   fun reasonable_date (date : int * int * int) =
  23       let
  24           fun get_nth (lst : int list, n : int) =
  25           if n=1
  26           then hd lst
  27           else get_nth(tl lst, n-1)
  28           val year  = #1 date
  29           val month = #2 date
  30           val day   = #3 date
  31           val leap  = year mod 400 = 0 orelse (year mod 4 = 0 andalso year mod 100
                        <> 0)
  32           val feb_len = if leap then 29 else 28
  33           val lengths = [31,feb_len,31,30,31,30,31,31,30,31,30,31]
  34       in
  35           year > 0 andalso month >= 1 andalso month <= 12
  36           andalso day >= 1 andalso day <= get_nth(lengths,month)
  37       end
```

Mark as completed