UCLA Computer Science 131 (spring 2019) midterm
100 minutes total, open book, open notes,
No computer or any other automatic device. Write answers on test.
Please be brief; excessively long answers will be penalized.

Name: David Hou   Student ID: 804995251

```
-----+-----+-----+-----+-----+-----+-----+
1    |2    |3    |4    |5    |6    |7    |total
     |     |     |     |     |     |     |
     |     |     |     |     |     |     |
-----+-----+-----+-----+-----+-----+-----+
```

1a (12 minutes). For each of the following OCaml function definitions, give the type of the function and explain in words what the function does, from the caller's point of view. Assume the usual environment where '*.' means 'float' multiplication as in (3.0 *. 4.0), and where 'sin' means the 'float' trigonometric function as in (sin 1.5).

let q f x = f x x

('a -> 'a -> 'b) -> 'a -> 'b

calls f (passed in) with 2 identical arguments specified by x.

let s = q (*.)

float -> float

squares a floating pt number.

let p a b x = a (b x)

('b -> 'c) -> ('a -> 'b) -> 'a -> 'c

calls a with argument of b called on x. p a b composes a and b. p a b x calls the composed function on x

let c = p s sin

float -> float

$c(x) = \sin^2(x)$,

1b (3 minutes). In 1a, why does s's definition use '(*.)' (1) and not '(fun x y -> x *. y)' (2) or '(*.)' (3) or simply '*.'? (4)

*. is an infix operator - (␣ *. ␣) clearly implies that its two parameters are absent. ((1) > (3))

(1) is more succinct than 3 and loses a function call (more efficient)

(4) will have the parser attempt to associate q as the first argument to the operator, and is confusing to read

2. Recall that the transpose of an M×N matrix A is an N×M matrix B such that A[i][j] = B[j][i] for 0≤i≤M, 0≤j≤N.
<   <

2a (12 minutes). Suppose we represent a matrix of items as a list of list of items. Write a function loltp that does list-of-list transposition, that is, it takes a list of list of values that represents a matrix A, and returns a list of list of values that represents the transpose of A. For example, (loltp [["a";"b";"c"];["d";"e";"f"]]) returns [["a";"d"];["b";"e"];["c";"f"]]. If you cannot reasonably solve the problem in general, make sure that it at least succeeds on a 2×3 test case such as in the example given, and explain why a more-general solution is not reasonable.

```
let popheads A = match A with
    | [] -> Some ([],[])
    | h::t -> match h with
          | [] -> None
          | h_::t_ -> match popheads t with
                | None -> None (* or assert false; matrix is not square *)
                | Some(res, rem) -> Some (h_::res, t_::rem);;

let loltp A = match popheads A with
    | None -> []
    | Some(col, rem) -> col :: (loltp rem);;
```

2b (2 minutes). What is the type of your loltp function?

'a list list -> 'a list list

2c (4 minutes). Give an example value that you can pass to loltp that OCaml's type checking will accept but will cause a runtime error; or explain why no such value is possible.

[[1;2;3];
 [1]]

will produce incorrect result:
[[1;1]]

3a (12 minutes). Suppose we instead represent a matrix of items in OCaml as a tuple of tuple of items. Write a function tottp that does tuple-of-tuple transposition; that is, it acts like loltp except it operates on the tuple-of-tuple representation. For example, (tottp (("a","b","c"),("d","e","f")) returns (("a","d"),("b","e"),("c","f")). Again, if you cannot reasonably solve the problem in general, make sure that it at least succeeds on a 2×3 test case such as in the example given, and explain why a more-general solution is not reasonable.

```
let tottp A =
    let ((a,b,c),(d,e,f)) = A in
        ((a,d),(b,e),(c,f)) ;;
```

More general impossible without breaking out of (OCaml) typesystem or doing clever hacky programming because general-length tuple types are not a part of OCaml's type system. (Also, no easy way to iterate a tuple)

3b (2 minutes). What is the type of your tottp function?

```
(('a * 'b * 'c) * ('d * 'e * 'f)) ->
                  (('a * 'd) * ('b * 'e) * ('c * 'f))
```

3c (4 minutes). Give an example value that you can pass to your tottp function that OCaml's type checking will accept but will cause a runtime error; or explain why no such value is possible.

Not possible, this function works on all 2x3 nested tuples, do not contain code that throws exceptions, and doo not assume anything about the argument value except that it is 2x3 nested tuple.

4. Given a grammar, a nonterminal symbol is called "nullable" if a valid parse tree rooted at the symbol contains no terminal symbols. For example, consider the following Homework 1 style grammar:

```
 let nullg =
   ["Expr", [T"("; N"Expr"; T")"];
    "Expr", [N"Expr"; N"Ops"; N"Expr"];
    "Expr", [T"ID"];
    "Ops", [N"Op"; N"Op"];
    "Ops", [N"Op"; N"Op"; N"Ops"];
    "Op", [T"+"];
    "Op", [];
    "Op", [T"*"]]
```

In this grammar, the nonterminal "Op" is nullable because it can produce the empty list immediately in the second-to-last rule, and the nonterminal "Ops" is nullable because it can produce two "Op" nonterminals, each of which can produce the empty list. However, "Expr" is not nullable.

4a (12 minutes). Write an OCaml function (nullables G) that returns the set of nullable nonterminals in the grammar G, representing the set as a list. The members of the returned list can be in any order and the list can contain duplicates. For examples, (nullables nullg) might return ["Ops";"Op"]. Your function can assume the functions (subset A B), (equal_sets A B), (set_union A B), (set_intersection A B), (set_diff A B), and (computed_fixed_point EQ F X) that were assigned in Homework 1. However, your function should not use any functions in the OCaml standard library. You can write auxiliary functions to help implement your function.

```
let contains l x = match l with
  | [] -> false
  | h::t -> if (h = x) then true else contains t x

let lall f l = match l with
    | [] -> true
    | h::t -> if (f h) then (lall f t) else (false);;
let lfilter f l = match l with
    | [] -> []
    | h::t -> if (f h) then h::(filter f t) else ( filter f t);;
let lmap f l = match l with
    | [] -> []
    | h::t -> (f h) :: (lmap f t);;
let transition rules nl = set_union (lmap (fun (nt, r) -> nt) (
                          lfilter (fun (nt, r) -> lall ( fun x ->
                                   match x with
                                   | T _ -> false
                                   | NT y -> contains nl y;;) r))) nl;;

let nullables rules = comput_fixed_point (equal-sets) (transition rules) [];;

(* sorry for the blob of list functions :( if you can match the parens, it
   should be clear. *)
```

4b (5 minutes). If you translate 'nullg' to Homework 2 style, will it cause the corresponding matcher to loop in some cases? If so, give an example of how the matcher would loop; if not, explain why not.

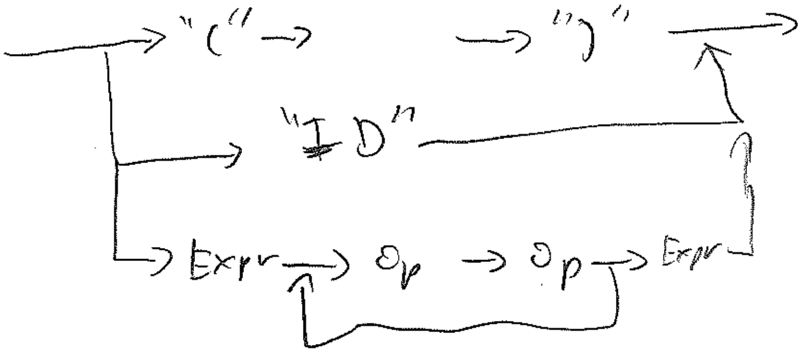
No, I treat this grammar as a function from NT -> rule lists always.

compute fixed point guarantees by resultlist (hl) changes every iteration until the end, and hl can only grow. By pigeon hole principle, it terminates. I do not make any recursive calls other than to iterate lists.

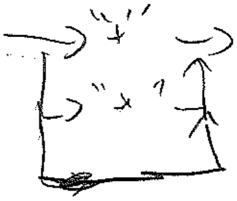
Getting rule (NT) is same process, can just translate in-function.

4c (10 minutes). Convert 'nullg' to a syntax diagram that is as simple as possible.

Expr:



Op:



5 (4 minutes). In OCaml, 'int list' is a subtype of ''a list'. However, in Java, 'List<Integer>' is not a subtype of 'List<Object>'. Explain the seeming discrepancy, and give some other List type that 'List<Integer>' *is* a subtype of in Java.

variables
A type can only be 'a list if everything that uses it
makes absolutely no assumptions about the type of the
things stored in the list ("enforced" by type inference). In
Java, no such restriction / enforcement exists, so one would be
able to add a String to a List<Integer> if it is subtype.
OCaml does not have this restriction since adding a string to the
'a list would have the list a string list instead - 'a list
describes that everything is OK for any type 'a.
List<Integer> subtype of List<Long> (never lose precision;
need cast though)

6 (6 minutes). In C, the _Noreturn keyword marks functions that do not return; for example, the 'exit' function is declared this way:

```
_Noreturn void exit(int);
```

because it accepts an integer argument and never returns. Compilers can optimize calls to _Noreturn functions, e.g., by generating code that does not bother to save the call's return address (because the return address is never used).

Is a function type containing the _Noreturn keyword a subtype of the same function type without _Noreturn? Or vice versa? Or neither? Briefly explain.

normal f is subtype of _Noreturn f. One can use a normal
f anywhere a _Noreturned function is used to the same effect,
just the program will have a better chance of continuing
to run. However, if _Noreturn guarantees that the function
will not return by itself, then neither is subtype - returning
function can continue and Noreturn function will quit the program
(while normal one won't, will affect results if there is code after the
function call.)

7 (12 minutes). The Java designers were willing to give up some performance in exchange for reproducible results. For example, although C allows a compiler to evaluate a call's arguments in any order, Java requires the compiler to evaluate them left-to-right. Java's rule prevents some optimizations but means that code is more likely to yield the same results on different platforms. A Java compiler is still allowed to execute the arguments out of order for speed, so long as the user can't tell the difference.

A significant reliability problem in Java comes from race conditions in multithreaded programs. Couldn't the Java designers have traded performance in exchange for avoiding race conditions? That is, couldn't the Java designers have said that a Java compiler must evaluate multithreaded code as if the first runnable thread is the only running thread? (By "first" I mean the earliest-created thread.) As before, the Java compiler would still be allowed to execute code in parallel for speed, so long as the user can't tell the difference other than in performance.

If this idea is impossible, explain why that is so. Or if it is possible but impractical, explain why. Or if it is a reasonably practical suggestion, give a good reason why the Java designers did not take the suggestion.

This is possible but probably impractical. (Rust has a similar idea - but they completely disallow behavior that could affect other threads in an unsafe way!) This is also just automatically enforced locking.

1. This would remove all CPU-performance benefit to multi threaded code if ~~[illegible]~~ unsafe code is running - but they should be using locks anyways, and this just acts like a lock on resources! However, this difficult to granularly share resources - we are in effect putting an ~~[illegible]~~ RW/ on everything. The detection of safety is a hard but not impossible problem. If code is [illegible], it will be single threaded, which will prevent very smart people from writing their own precarious but safe parallel code. Algorithms to detect "stepping on toes" do exist, but they are overly cautious and impose big restrictions.

2. If the first thread (running) is CPU-blocked but others have IO tasks to do, this scheduler causes massive latency. This is especially important in server software - the latency can become unbearable. Python GIL is a less safe version of this - it has a more dynamic scheduler though, so this [illegible]

3. This doesn't completely guarantee data race safety - races could still happen across yields! ~~It is controlled though~~ The places races could happen is controlled though!

4. Complex compiler code.

Rust effectively does this, but at compile time, and in a better way! ~~This is to~~ It imposes restrictions though, on unknown accesses [illegible]