CS 131 Midterm

TOTAL POINTS

78 / 100

QUESTION 1

camp types 35 pts

1.1 12 / 12

- **+ 0 pts** Blank/ All wrong
- ✓ **+ 1 pts** **x correct in first**
- ✓ **+ 1 pts** **f correct in first**
- ✓ **+ 1 pts** **q correct**
- ✓ **+ 1 pts** **s correct**
- ✓ **+ 1 pts** **b correct**
- ✓ **+ 1 pts** **a correct**
- ✓ **+ 1 pts** **p correct**
- ✓ **+ 1 pts** **c correct**
- **+ 1 pts** Insufficient explanation
- **+ 2 pts** Explanations partly correct.
- **+ 3 pts** Explanations mostly correct. Minor issues
- ✓ **+ 4 pts** **All explanations correct**

1.2 2 / 3

- **+ 1 pts** Correct reasoning for fun
- ✓ **+ 1 pts** **Correct Reasoning for (*.)**
- ✓ **+ 1 pts** **Correct reasoning for *.**
- **+ 0 pts** No correct answer

QUESTION 2

Matrix transpose with list of lists 18 pts

2.1 12 / 12

- ✓ **- 0 pts** **Correct**
- **- 1 pts** Very small syntax mistakes
- **- 2 pts** Minor mistakes in logic
- **- 2 pts** Minor mistake in syntax
- **- 4 pts** On the right track but missing few things/ some things wrong
- **- 6 pts** Partially correct
- **- 8 pts** Explanation for no general solution is unreasonable (This is because general solution is easily possible)
- **- 8 pts** Incomplete solution/ gave 2xn or 2x3 solution with no explanation
- **- 9 pts** Only gave pseudo code/ explanation on how to solve. i.e incomplete solution.
- **- 9 pts** Mostly incorrect
- **- 12 pts** Wrong
- **- 12 pts** No answer

2.2 2 / 2

- **- 0.5 pts** minor mistake
- **- 1 pts** 1 of input or output type is wrong
- **- 1.5 pts** Mostly incorrect
- **- 1.5 pts** Not in ocaml format
- **- 2 pts** Wrong
- **- 2 pts** No answer
- ✓ **- 0 pts** **Correct**

2.3 0 / 4

- **- 0 pts** Correct; Input is 'a list list but not a matrix
- **- 2 pts** example not complete
- ✓ **- 4 pts** **Wrong**
- **- 4 pts** No answer

QUESTION 3

Matrix transpose with tuple or tuples 19 pts

3.1 10 / 12

- **- 0 pts** Correct
- **- 12 pts** Incorrect
- **- 6 pts** Partly correct
- ✓ **- 2 pts** **Minor problems**
- **- 9 pts** Mostly incorrect
- **- 3 pts** Small mistakes

- **- 9 pts** Some right ideas
- **- 10 pts** Mostly incorrect
- **- 1 pts** Minor mistakes
- **- 4 pts** Need better reasoning for why general case not possible
- **- 2 pts** Explanation unclear
- **- 2 pts** Minor problems with explanation
- **- 6 pts** Explanation for general case missing
- **- 3 pts** Needs better explanation for why general case not possible
- **- 4 pts** Problems with explanation

3.2 2 / 2

✓ **- 0 pts Correct**
- **- 2 pts** Incorrect
- **- 1 pts** Partly correct
- **- 1.5 pts** Mostly incorrect
- **- 0.5 pts** Minor mistakes
- **- 1.5 pts** Some right ideas

3.3 4 / 4

✓ **- 0 pts Correct**
- **- 4 pts** Incorrect
- **- 2 pts** Partly correct
- **- 1 pts** Minor mistakes
- **- 1 pts** Needs more explanation
- **- 2 pts** Needs better explanation
- **- 3 pts** Missing explanation
- **- 3 pts** Mostly incorrect

QUESTION 4

Grammar 17 pts

4.1 9 / 12
- **- 0 pts** Correct
- **- 12 pts** Empty
- **- 1 pts** Missing rec keyword when defining recursive funcs
- **- 2 pts** use of any functions in the OCaml standard lib is not allowed
- **- 1 pts** Wrong return type

✓ **- 3 pts Partially correct code I**
- **- 5 pts** Partially correct code II
- **- 7 pts** Partially incorrect code: III
- **- 9 pts** Code only iterates over rules
- **- 7 pts** Incomplete code : I
- **- 9 pts** Incomplete code: II
- **- 11 pts** Incomplete code: III

💬 problematic test_LOR function, which only returns true.

4.2 3 / 5
- **- 0 pts** Correct
- **- 5 pts** Empty
- **- 1 pts** No example I
- **- 2 pts** No example II
- **- 3 pts** No example III
- **- 3 pts** No explaination

✓ **- 2 pts Incorrect answer, but with some justificati**
- **- 4 pts** incorrect answer and justification
- **- 2 pts** Correct answer & Incorrect argument
- **- 5 pts** Incorrect answer no justification

4.3 10 / 10

✓ **- 0 pts Correct. Well Done!**
- **- 2 pts** No Optimization. Represent using only Expr.
- **- 1 pts** Incorrect rule
- **- 2 pts** Incorrect rules
- **- 10 pts** Unattempted
- **- 5 pts** Ops, Op missing
- **- 1 pts** Click here to replace this description.
- **- 4 pts** Incorrect rules
- **- 1 pts** Incorrect optimization
- **- 9 pts** wrong
- **- 1 pts** Merge into expr
- **- 1 pts** Summarized poorly
- **- 7 pts** Wrong
- **- 1 pts** Represent using only Expr. More optimization required.
- **- 0 pts** Click here to replace this description.
- **- 3 pts** incorrect rules
- **- 5 pts** missing rules
- **- 3 pts** Incomplete

- **- 8 pts** wrong
- **- 5 pts** Incorrect representation

QUESTION 5

5 java/ocaml Subtyping 3 / 4

- **- 0 pts** Correct
- **- 4 pts** Unattempted
- ✓ **- 1 pts** **No mention of Generics.**
- **- 1 pts** Wrong example.

List <Integer> is a subtype of List <?>, Collection <Integer>

- **- 4 pts** Wrong
- **- 1 pts** Unclear argument
- **- 1 pts** example missing.
- **- 1 pts** Incorrect statement: Integer is not primitive.
- **- 1 pts** Integer IS a subtype of Object. Wrong statement.
- **- 1 pts** Incomplete argument
- **- 1 pts** Partially correct argument
- **- 1 pts** Missing explanation
- **- 1 pts** Click here to replace this description.
- **- 3.5 pts** Incomplete
- **- 1 pts** Integer is not an instance of Object, it is a subtype
- **- 2 pts** missing explanation

QUESTION 6

6 C subtype for _Noreturn 3 / 6

- **- 0 pts** Correct
- **- 6 pts** Unattempted
- ✓ **- 3 pts** **Correct ans: No_return is a subtype**
- **- 6 pts** Wrong
- **- 4 pts** unclear argument
- **- 2 pts** Partially correct
- **- 5 pts** Doesn't make sense
- **- 4 pts** Incorrect
- **- 1 pts** Better explanation required.
- **- 2 pts** Better explaination required
- **- 5 pts** Missing explanation
- **- 0 pts** Click here to replace this description.

QUESTION 7

7 Java design question 6 / 12

- **+ 0 pts** Black answer/ Fully incorrect/ No explanation
- **+ 2 pts** Significantly insufficient argument
- **+ 4 pts** Insufficient arguments/ not completely in right direction
- ✓ **+ 6 pts** **Correct answer/Arguments not developed/Incorrect answer/ Arguments developed in sort of right direction.**
- **+ 8 pts** Correct answer. Arguments somewhat developed/ not fully correct.
- **+ 10 pts** Correct answer. Argumentsmostly developed.
- **+ 12 pts** Correct answer. Arguments correct and completely developed.

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: _____ Student ID: _____

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
12 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

this function takes a function f and and argument x and returns the value of calling the function w/ two arguments, both of which are x

```
let s = q ( *. )
```
float -> float

this function is a curried form of q that takes a float and returns the float value of it squared

```
let p a b x = a (b x)
```
('a -> 'b) -> ('c -> 'a) -> 'c -> 'b

this function takes two functions a and b and an argument x and returns the composition of these two functions

```
let c = p s sin
```
$s(\sin(x)) = [\sin(x)]^2$

float -> float

this function is a curried function that takes an argument (not shown) and first computes its sin and then squares it.

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

parentheses are required to let it know it is talking about the FP binop and not actually using it so '*.' cannot work. '(*.)' cannot work b/c '(*' is used to start comments. Using a lambda function would work on its own, but it would cause problems when curried into q.

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.

27 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 rec get_first_col = function
  | [] -> []
  | frow :: rest ->
      match frow with
        | [] -> []
        | hd::tl ->
            hd :: get_first_col rest
```

item → "a"; "d"

inputs

helper function that takes a list of lists and returns a list of the first item in each list

```
let rec loltp = function
  | [] -> []
  | M ->
      let rest = List.map (List.tl) M in
      let curr = get_first_col M in
      curr :: (loltp rest)
```

Gets new row by getting first item in each row. Then removes first item from each row and passes this back in recursively.

29 2b (2 minutes). What is the type of your loltp function?

'a list list → 'a list list

33 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.

This function works with the generic type 'a. That being said if the type used is not compatible with the cons (::) operation, this will cause a runtime error.

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.

US

Specific solution for 2x3:

```
let loltp matrix =
    let frow = fst matrix in
    let srow = snd matrix in
    let a = fst frow in
    let b = snd frow in
    let c = trd frow in
    let d = fst srow in
    let e = snd srow in
    let f = trd srow in
    ((a,d),(b,e),(c,f))
```

Tuples cannot be generalized like Lists b/c they have a fixed type which determines their fixed size. In order to generalize this function, it would require to accept/return different types (b/c of different sizes) which is not allowed

a b c
d e f

a d
b e
c f

47. 3b (2 minutes). What is the type of your tottp function?

('a * 'b * 'c) * ('d * 'e * 'f) ⟶ ('a * 'd) * ('b * 'e) * ('c * 'f)

Cannot assume each element is of same type.

51 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.

There is not such value b/c all it does is rearrange elements in tuples which are heterogenous, and my solution above only works for the 2x3 case so any input of different size may not run properly or cause a crash.

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.

63 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 rec get_rules g nt = match g with
  |[] -> []
  |(n,rule)::rest ->
    if n = nt then
      rule :: get_rules rest nt
    else
      get_rules rest nt
```

```
let rec has_t = function
  |[] -> false
  |hd::tail ->
    match hd with
      |(N n) -> has_t tail
      |(T t) -> true
```

```
let rec has_emp_rule = function
  |[] -> false
  |hd::tl ->
    if hd = [] then true
    else has_bad_rule tl
```

```
let rec has_bad_rule g nt =
  let LoR = get_rules g nt in
  if has_emp_rule LoR then true
  else
    test_LoR g LoR
and rec test_LoR g = function
  |[] -> true
  |hd::tl ->
    match hd with
      |(T t) -> true
      |(N nt) -> has_bad_rule g nt
```

```
let get_bad_nts x = match x with
  |(g, list_of_bad_nts) ->
    match g with
      |(n,r)::rest ->
        if has_bad_rule g n then
          (g, n::list_of_bad_rules get_bad_nts g rest)
        else
          (g, list_of_bad_rules get_bad_nts g rest)
```

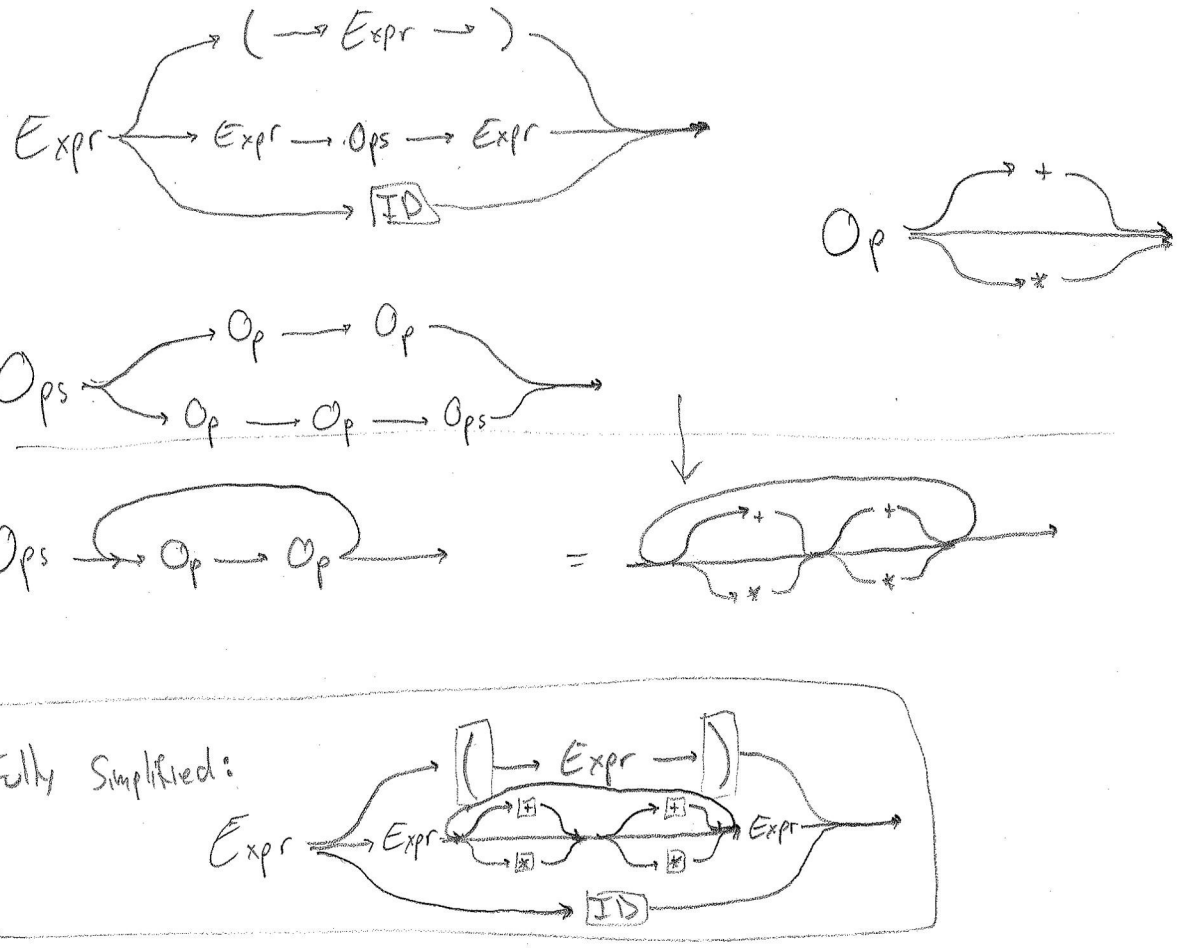
```
let nullables G =
  let nulls = computed_fixed_point
    (equal_sets)
    (get_bad_nts G[])
  in
  nulls
```

get_bad_nts returns g * list of all bad nts that either have bad rule ([]) have a lst w/ only bad nonterminals in them. When calling this w/ computed_fixed_point the final result will be the list of all bad, nullable nonterminals

4b (5 minutes). If you translate 'nullg' to Homework 2 style, will
68 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 implemented my approach the same either way since I just create my own production function get_rules which would be the same regardless of whether it came from my grammar or from my auxiliary functions. If there is a situation that loops it would happen ~~with~~ regardless of the form of grammar w/ my matcher.

78 4c (10 minutes). Convert 'nullg' to a syntax diagram that is as simple as possible.



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.

82

The seeming discrepancy comes from the difference between 'a and Object. By distinguishing the type of the list in OCaml the object gets added features whereas by indicating the list has type List<Integer> in Java it restricts the amounts of things that can be done b/c now it can't add in Objects to the list without causing a type error.

List<Integer> is a subtype of Collection<Integer> b/c it expands on the abilities of Collection<Integer>

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

88

```
_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.

A function with a return value would be a subtype of the _Noreturn function because it expands on the _Noreturn function's abilities by adding a return value to its member variables and method implementation.

for example:

```
class NoRetFunc {
    function
} ---

class RetFunc extends NoRetFunc {
    function
    return_value
} ---
```

100 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.

Yes. This is possible because race-conditions vs performance is a trade-off. They definitely could have allowed compilers to evaluate multi-threaded code as if the first thread was running alone to boost performance but this would cause problems the JMM and lead to race-conditions. Overall, this is impractical. No one would want to use a machine language that runs very fast with tons of race-condition related errors. This is especially true for Java as it is utilized for its reproducible results on many machines so to design a language like Java that runs on a portable VM, but that is not reliable due to race conditions just wouldn't make practical sense.