CS 131 Midterm

Siddharth Joshi

TOTAL POINTS

77.5 / 100

QUESTION 1

camp types 15 pts

1.1 8 / 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 4 / 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 18 pts

3.1 12 / 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 1.5 / 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 3 / 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 27 pts

4.1 6 / 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

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 justification
- **4 pts** incorrect answer and justification
- **2 pts** Correct answer & Incorrect argument
- **5 pts** Incorrect answer no justification

4.3 7 / 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

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 6 / 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 8 / 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: SIDDHARTH JOSHI Student ID: 105032378

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

```
let s = q ( *. )
```

'float → 'float

```
let p a b x = a (b x)
```

('a → 'b) → ('c → 'a) → 'c → 'b

```
let c = p s sin
```

'float → 'float

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

in 1a) the use of (*.) is to have the *. operator (that is infix) be passed to q as a function. (fun x y → x *. y) would technically be equally valid but is unnecessarily verbose. Moreover *. simply *. would result in an error as that usage ⇒ it is an infix operator for floats ∴ needs floats to either side.

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 rec loltp matrix =
  if List.flatten matrix = [] then []
  else
  let col = List.map (fun x -> List.hd x) matrix and
  let rest = loltp (List.map (fun x -> List.tl x) matrix) in
  if rest = [] then col
   else col :: rest
```

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.

For my implementation of loltp to work all the lists inside the main list should have equal size as while it is possible for a list list to have lists of different sizes (and some OCAML type checking will accept it), the implied matrix

would have unequal rows/cols which would make no sense. ∴ while such a value can trigger a runtime error ideally we would never receive such a value if the original 'matrix' is correct.

general not possible

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 matrix =
let third = function (a,b,c) -> c in
(((fst (fst matrix)),(fst (snd matrix))),
 ((snd (fst matrix)),(snd (snd matrix))),
 ((third (fst matrix)),(third (snd matrix))))
```

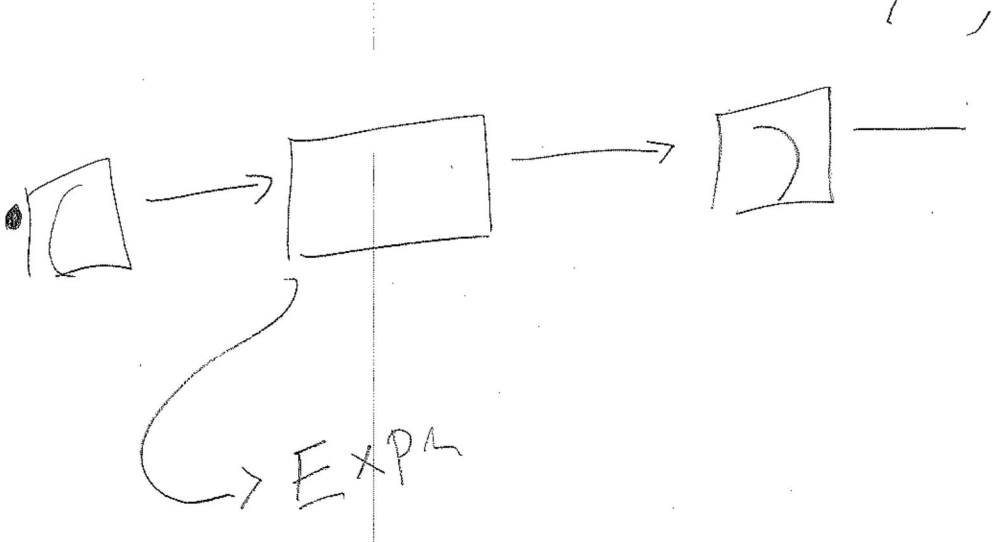
A general function is not possible here as tuples are of fixed sizes and there is no general way (as far as I know) to define an argument for a function that takes a tuple w any # of elements

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

('a * 'a * 'a) * ('a * 'a * 'a) -> ('a * 'a) * ('a * 'a) * ('a * 'a)

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.

No value is possible as such OCAML's type checker would ensure that the tuple implied matrix has dimension 2×3 (in particular a tuple w 2 elements that are both 3-element tuples) and that the types of element in the inner tuple match up (guaranteed by type var 'a)


Expr

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.

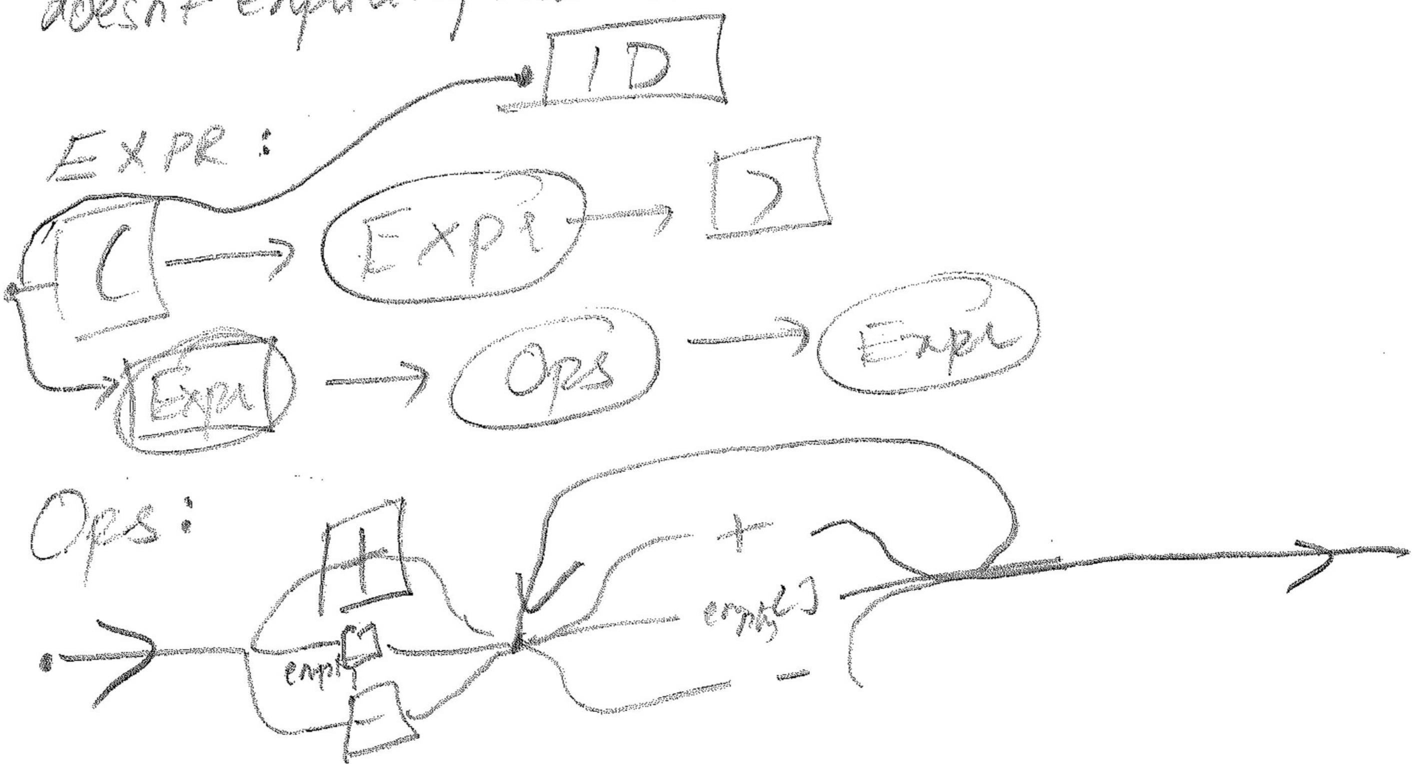
```
let find_nullables = function
| (_, rules) ->
let nullt = List.filter (fun x -> (snd x) = []) rules in
let null = List.map (fun x -> fst x) tuples in
let f set l = List.filter (fun x -> (set_diff (snd x) set) = []) l
in let rec g setl = non (f set l) in
(compute_fixed_point equal_sets (g null) rules
```

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.

It won't work as it assumes tuples depict a rule rather than an actual production function. Also if the impl were changed accordingly still there may be same loops depending on recursion of grammar

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

Assuming that expr is the start symbol as nullg itself doesn't explicitly indicate this



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.

List<Integer> is not a subtype of List<Object> as it is possible to add an element of type Thread to a List<Object> ∴, but the same functionality is not offered by List<Integer> ∴ since it doesn't support a superset of the functionality of List<Object> as subtypes must (List<Integer>)
(Interesting the converse is false as well List<Object> is not a subtype of List<Integer> this happens due to the violation of the condition that a subtype must only have a subset of values but List<Object> can take values that List<Integer> can't.
List<Integer> is a subtype 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:

```
_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.

I would argue that a function type containing the _Noreturn keyword is a subtype of a function type without it as the former can only accept functions that do not return but the latter can accept both functions that do return and those that don't. In particular a proof for this would be Moreover the functionality of the 2 types is identical as while _Noreturn is a hint to the compiler to optimize by not bothering to save the call's return address, it can be said compiler can very well carry out this optimization as well. This equality implies that the functionality subtype _Noreturn is the trivial superset (equal to the set) ∴ _Noreturn func type ≤ a subtype of w/o no return

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 idea is incredibly impractical as often it is possible in multithreaded applications for a thread's work to be dependent on the work done by some other thread. Imagine if the first thread to run were to carry out some ops and then wait

This idea while possible is extremely impractical as it would eliminate any parallelism the multithreaded program attempted to achieve as it is very likely that the work even in a simple case this would be equivalent to a sea sequential execution of the program. Moreover if threads could one created either complete execution or be in a WAITING state and resultantly all the code would be executed sequentially. I believe it so the entire idea of multithreading is to allow for performance gains by having "concurrent" execution but such a setup would eliminate that & make multithreading in JAVA useless (rather slower than

regular execution due to the overhead of thread creation, context switches etc.)

However race conditions are not the only bug that multithreading causes other issues like deadlock (if one thread is ~~called~~ is waiting for another ... cyclic dependency) could still occur in this system.

Thus the Java designers chose to not take such an approach