UCLA CS 131 Midterm, Winter 2018
100 minutes total, open book, open notes
closed computer. Exam is DOUBLE SIDED.

Name: Shrey Kokkar

1	2	3	4	5	6	total
7	2	1	8	24	18	

Student ID: 204792652

1. Rewrite each of the following OCaml definitions to an equivalent form by adding trailing integers to each identifier, using as many distinct integers as possible. Use the integers in order 0, 1, 2, ... For example, for 'fun a -> fun b -> (fun a -> a) (a + b)' you would write 'fun a0 -> fun b1 -> (fun a2 -> a2) (a0 + b1)'. Or, if it's not possible to rewrite as requested, explain why not.

```
1a (2 minutes).  let rec f x = f x
1b (3 minutes).  type ('nonterminal, 'terminal) symbol =
                    | N of 'nonterminal
                    | T of 'terminal
1c (4 minutes).  let a a a = function |(_,a,_) -> a
```

2. Convert each of the definitions (1a), (1b), (1c) into a simple form with no shorthand. In the simple form, every 'let' should be of the form 'let ID = EXPR', every lambda expression should be of the form 'fun ID -> EXPR', where ID stands for a single identifier and EXPR for a single expression. Or, if it's not possible to rewrite a definition as requested, explain why not. (Problem values are the same as for problem 1.)

3. For each of the definitions (1a), (1b), (1c), list the types of every top-level identifier. (An identifier is "top-level" if it is visible to later definitions in the same program.) Or, if it's not possible to list a top-level identifier's type, explain why not. (Problem values are the same as for problem 1.)

4 (15 minutes). Consider the following Java class:

```
class Rebar {
  int n = 0;
  volatile boolean v = false;
  int foo() { return v ? n : -1; }
  void bar() { if (n != -1)
                 n++;
               v = true; }
}
```

In a sequential program, 'foo' must always return nonzero. However, can 'foo' return zero in a multithreaded program? If so, give a scenario where this can occur, and add the appropriate synchronization primitives to the Rebar class so that 'foo' always returns nonzero even in a multithreaded program; if not, explain why not and justify your answer by appealing to the JMM.

Shrey

let

2

b)

5. Consider the following grammar for a subset of the conents of "From:" lines in email, taken from Internet RFC 5322 and simplified somewhat. The start symbol is "mailbox-list".

```
mailbox-list    =   mailbox *("," mailbox)
mailbox         =   addr-spec / angle-addr
angle-addr      =   "<" addr-spec ">"
addr-spec       =   local-part "@" domain
local-part      =   dot-atom / quoted-string
domain          =   dot-atom
quoted-string   =   DQUOTE *(qcontent) DQUOTE
qcontent        =   qtext / quoted-pair
qtext           =   atext / "[" / "]" / "," / "<" / ">" / "@" / "."
quoted-pair     =   "\" anychar
anychar         =   qtext / "\" / DQUOTE
dot-atom        =   1*atext *("." 1*atext)
atext           =   ALPHA / DIGIT / "*" / "+" / "-" / "/"
```

> c

@

/ [ ] \

@ . + -

5a (6 minutes). What are the tokens of this grammar?

5b (2 minutes). What are the nonterminals of this grammar?

5c (10 minutes). Prove that the grammar is unambiguous.

5d (10 minutes). Translate this grammar to BNF. Make as few changes as possible. Write your BNF in the style of RFC 5322.

5e (10 minutes). If you took the BNF version of this grammar, converted it to a form suitable for Homework 2, and submitted it to a correct solution to Homework 2, could that cause an infinite loop? Briefly explain.

6 (20 minutes). Would it make sense to write a compiler to translate C source code to Java bytecodes? The idea is to run your C program on a Java interpreter; your program would be accompanied by an implementation of the C library written in a combination of Java and machine code, just as the traditional C libary is implemented in a combination of C and machine code. If it would make sense to write the compiler, list any difficulties you'd have in writing it or the associated library, and list practical pros and cons of the resulting system compared to the traditional approach. If it would not make sense, explain why not, and list the features of C that you'd need to drop support for, in order to make the job practical. When answering the question, consider all the Java features covered in class.

Shrey Kokkar 204792652

a) let rec fo x1 = f0 x1

2

b) type ('nonterminal, 'terminal) symbol =

1

| NO of 'nonterminal

| T1 of 'terminal

c) let a0 a1 a2 = function | (_, a3, _) -> a3

4

Q2

a) let rec x = (fun f -> (f x)) f

0

b) Cannot be rewritten in Simple form. Because of tuple type, defining both nonterminal & terminal is required for the code to compile. we need to give proper explanation of type (Eg N of 'nonterminal) for both nonterminal & terminal.

1

c) let a0 a1 = (fun a2 -> (a0 a1)) -> (fun a3 -> (a2 a1)) a3

1

can't ~~[illegible]~~ name? 3 ~~[illegible]~~ args in let. must

directly return fn.

match?

Currying?

Q3 → a) The top level identifiers here are

A valid, albeit not the most general, type

+1

f → $\boxed{\text{'a} \rightarrow (\text{a'} \rightarrow \text{a' list})}$ = < fun >

b) top level identifier here are 'non ~~ter~~minal and 'terminal

type ('nonterminal, 'terminal) symbol = N of 'nonterminal | T of 'terminal

c) top level identifier here is a

a: a' list → b' list → (a' list → ~~b' l~~ist → b') → a' = < fun >

Shrey Kokkor 204792652

1 → I think that 'foo' CAN return Zero in a multithreaded program (NOTE v is volatile! → so no cache storing memory)

Consider this case

```
Thread A              | Thread B              | Thread C
                      |                       |
void bar()            | int foo()             | void bar()
  (if n!=-1)          |                       |   (if n!=-1)
     v = true;        |  return v? n:-1;      |      n++;
```

In this task distribution, if Thread A runs first, B second & C third,

So, Thread A first find n=0 (whence n!=-1 == true), and thus sets v=true;

now Thread B reads this new value of v(true) & hence returns n (still 0)

now thread C sees (n!=-1) and sets n++;

So, foo returned zero!

→ Adding synchronization!

```
class Rebor {
   int n=0;
   booleon v=false;
   syncronized int foo() { return v?n:-1; }
   Synchronized void bar() { if (n!=-1)
                              { n++;
                                v=true; }

}
```

Since all functions are Synchronized, only 1 thread can run them.

Hence at all times foo will return a non zero value.

Q6/1

~~Yes~~, I think that it is practical to write a compiler to translate C source code to Java byte codes. There would be a lot of difficulties and a lot of cons to it, but still it would work.

Some problems that might come up are that C right now has a lot of intermediate steps (foo.c → foo.i → foo.s - - - -). If converting to byte code method, we will have to get rid of these ~~to~~ kinds of intermediate steps. Also, we will have to find a way to strip down the source code. This will, however, give us a better security of code

The pros to this would definitely be security. Seeing that C is already very portable, that wouldn't be too much of an added pro.

The ~~cons~~ biggest cons would be Speed! Bytecodes are slower. So we will lose on some speed perspective.

Another con would be division of code. Now, we can just remove a piece (part) ~~of~~ in C, and switch it with something else (Distribution of work). This would be lost on converting to byte code methodology.

HREY KAKKAR (204 792 652)

5a) The tokens are:) < > @ [ ] . \ * ALPHA DIGIT + - / DQUOTE

+6 ✓

5b) The nonterminals of this grammar are :-

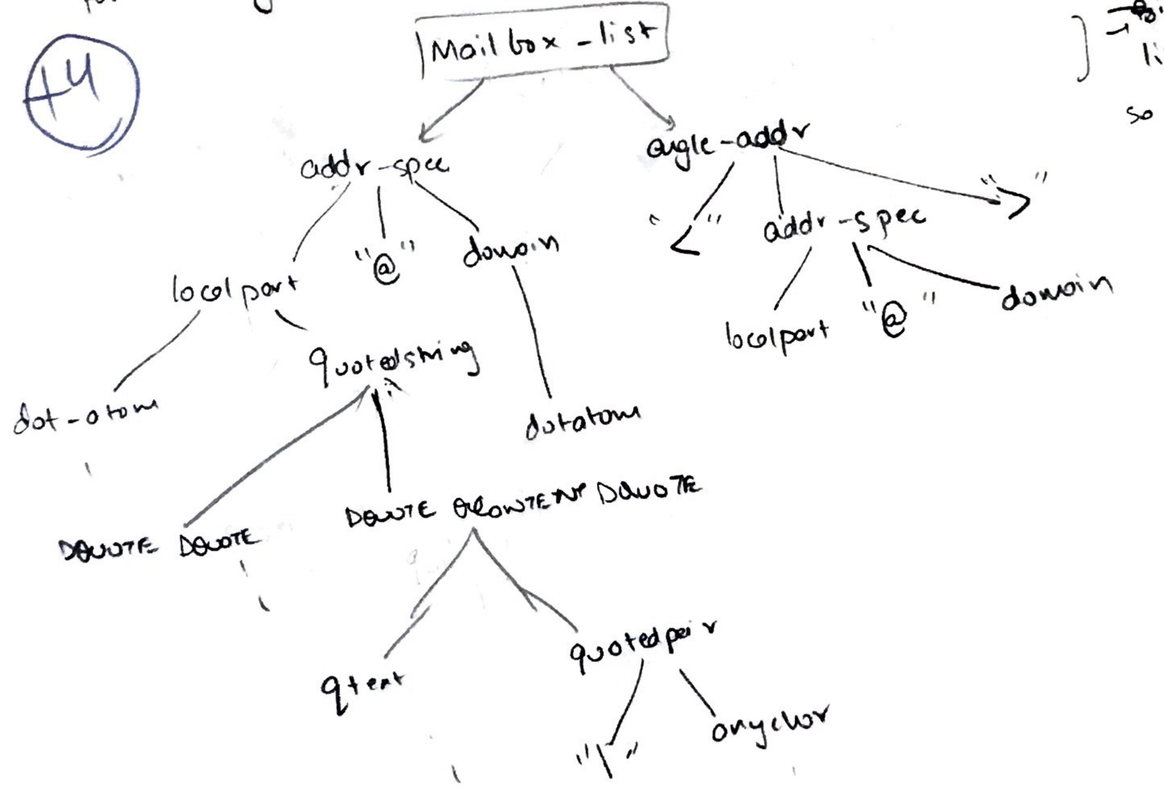
+2

mailbox-list | mailbox | angle-addr | local-part | domain | quoted-string | qcontent | qtext | quoted-pair | any char | dot-atom | atext | addr-spec

5c) To prove grammar is unambigous, we need to show that all parts in the grammar are segregated (i.e-nothing can have 2 meanings) ✓. This can be shown by forming a tree for the grammar. If there is only one possible tree for this grammar, then it is unambigous

X

+4



] ~~To~~ Angle-addr is a list of addr-spec, so it being after addr-spec makes grammar unambigous

X

we can see that in no form can this tree be drawn in a different way, as everything is well specified. Hence the grammar is unomligous

```
5d) <mailbox-list> ::= <mailbox> ✓ | <mailbox> "," <mailbox-list>
6   <mailbox> ::= <addr-spec> | <angle-addr>
    <angle-addr> ::= "<" <addr-spec> ">"
    <addr-spec> ::= <local-part> "@" <domain>                     -2
    <local-part> ::= <dot-atom> | <quoted-string>
    <domain> ::= <dot-atom>
    <quoted-string> ::= DQUOTE DQUOTE | DQUOTE <qcontent> DQUOTE | DQUOTE <qcontent> <quoted-string> DQUOTE
    <qcontent> ::= <qtext> | <quoted-pair>
    <qtext> ::= <atext> | "[" | "]" | "," | "<" | ">" | "@" | "."
    <quoted-pair> ::= "\" <anychar> | "\" | DQUOTE
    <anychar> ::= <qtext> | <atext> <dot-atom> |
    <dot-atom> ::= <atext> "." <atext> | <atext> "." <dot-atom>    -2
                                                                   <atext>
    <atext> = ALPHA | DIGIT | "*" | "+" | "-" | "/"
```

5e) If BNF version of this grammar was put to HW2, it CANNOT cause an infinite loop. This is because there is no single that keeps calling on itself. By this, I mean by assuming/following no path will you ever enter a situation where there is no "eventually ending path" available.

6

This can allo be explained by the fact that even if we find a blind-alley, there is a rule that exists before the blind alley, that is not a blind-alley.

?