COM SCI 131 Midterm

Sriram Balachandran

TOTAL POINTS

78 / 100

QUESTION 1

1 1 7 / 8

+ **0 pts** Incorrect

+ 7 Point adjustment

QUESTION 2

2 2 2 / 10

+ **0 pts** Incorrect

✓ **+ 2 pts Only signature is correct**

+ **5 pts** Basic idea is correct but it does not work. Be aware that the first nt should be handled differently.

+ **8 pts** Minor issues only

+ **10 pts** Correct

QUESTION 3

3 20 pts

3.1 3a 10 / 10

✓ **- 0 pts Correct**

- **5 pts** Incorrect reasoning for lo=ls
- **3 pts** Right idea, needs more explanation
- **3 pts** lo=ls needs more explanation
- **5 pts** Partly correct
- **2.5 pts** Needs explanation for why unmodifiableList works (extends keyword)
- **2.5 pts** lo=ls partly correct
- **5 pts** unmodifiableList not explained
- **8 pts** Some right ideas, but missing the main points
- **2 pts** unmodifiableLIst needs more explanation
- **3 pts** Needs more explanation for unmodifiableList
- **3 pts** No mention of List<String> not being a subtype of List<Object>
- **5 pts** Incorrect reasoning for unmodifiableList
- **2.5 pts** Some mistakes in unmodifiableList explanation
- **1 pts** Minor mistakes in lo=ls
- **2 pts** Minor mistake in unmodifiableList
- **10 pts** Incorrect

3.2 3b 10 / 10

✓ **- 0 pts Correct**

- **8 pts** Incorrect reasoning
- **6 pts** Needs more explanation
- **4 pts** Needs to explain what T could be
- **4 pts** Needs an example of how T could be multiple different types
- **10 pts** Incorrect
- **5 pts** Some mistakes
- **4 pts** Some right ideas but unclear reasoning
- **3 pts** Right idea, some mistakes
- **7 pts** Unclear reasoning
- **2 pts** Right idea, some mistakes

QUESTION 4

4 12 pts

4.1 4a 3 / 3

The correctness of type

✓ **+ 1.5 pts components types written down are correct: 'a -> ('a -> 'b) -> 'b**

✓ **+ 1 pts correct amount of components (3: typeof(f) -> typeof(g) -> typeof(rah))**

✓ **+ 0.5 pts using polymorphism**

+ **0 pts** None of the above

+ **0.5 pts** Some correct explanations on its value, figuring out we are asking for g(f) (bonus)

- **0.5 pts** completely correct in theory but some typos (e.g. introducing unneeded 'c to replace some occurrence of 'b; but using 'a to replace 'b is not considered a close-hit --- by saying two 'a you are

expecting them having to be the same type!) (if there's no 'b, only 'a and 'c, it is also okay... could be observed in practice anyway)

- 0.5 pts almost everything correct (only applies to the answer when otherwise the answer is full score) but having the wrong conclusion that this function is buggy (it is not, please try it in OCaml if you have questions on it).

4.2 4b 2 / 3

The correctness of type

✓ **+ 0.5 pts all details correct: (string * string list) list**

✓ **+ 0.5 pts recognizing the tuples**

✓ **+ 0.5 pts recognizing the lists (inner and outer, both, together)**

✓ **+ 0.5 pts recognizing the strings**

+ 0 pts no clue indicating correct understanding, or missing type

The correctness of value

+ 0.5 pts all types applied correctly in the value

+ 0.5 pts all values filled-in correctly (e.g. expr to "3+4")

✓ **+ 0 pts overall value not correct or missing a valid overall value**

+ 0.5 pts (bonus) correct and detailed description, in addition to some correct conclusion

- 0.5 pts typo in expression (such as , instead of * for tuple is incorrect; view it as a function when it shouldn't be)

4.3 4c 4 / 6

The correctness of type (ground-truth answer is: ('a -> 'a list -> 'b option) -> 'a -> 'a list -> 'b option)

✓ **+ 1 pts Figuring out that a type contains 3 components**

✓ **+ 1 pts Figuring out that n type contains 1 component**

✓ **+ 1 pts Figuring out that the output value is a function with 1 argument, and know that the argument is not n**

✓ **+ 1 pts Write-down the type of each component generally correctly (relaxations on Option: knowing and indicating it is option is enough in this part; relaxation on insevere typo: if you have other typos but generally correct you don't get the next point, but you still have this point)**

+ 1 pts Considering all components, the types are also correct, at least know an option it is an option (which means that there's no 'a and 'b messed up etc.)

- 1 pts Having problem expressing the correct type of Option.

+ 1 pts Having the correct value (defined this way, the value is the output function / the output of the output function) (note: only indicating that it is a function is NOT enough!)

+ 0.5 pts (bonus) Having some intermediate steps for reasoning, or reasonable explanations.

+ 0 pts Not knowing the answer at all / leaving the answer blank

QUESTION 5

5 20 pts

5.1 5a 7 / 12

- 0 pts Correct

- 3 pts C2 is not safe

- 3 pts Justify why C0 and C2 are unsafe

- 2 pts Justification missing. No mention of optimizations (loads,stores,reordering)

- 3 pts C3 is safe

✓ **- 3 pts C1 is safe**

- 6 pts class 1 is safe. Class 2 is not reliable.

- 9 pts No mention about C0,C1,C2

- 6 pts partially correct

✓ **- 2 pts Justification of C0 missing**

- 9 pts no mention of C1,C2,C3

- 8 pts Havent mentioned what is safe and what is not

- 1 pts C2 is not reliable

- 4 pts No explanation

- 1 pts justification for C0 missing

5.2 5b 8 / 8

✓ **- 0 pts** **Correct**

- 4 pts Incorrect ordering. C0>C2>C1>C3

- 4 pts No explanation

- 2 pts C0 missing

- 2 pts justification not clear

- 3 pts Unclear between C1 and C2

- 3 pts partially correct

- 6 pts Wrong

- 2 pts C1 is faster than C3

- 2 pts C1 missing

QUESTION 6

6 15 pts

6.1 6a 4 / 5

✓ **+ 1 pts** **Show understanding of ambiguity**

✓ **+ 1 pts** **Give an example correctly contains ambiguity & syntactically correct**

✓ **+ 2 pts** **Illustrate the two different ways**

+ 1 pts Explain the ambiguity by drawing two different parse trees, or explain it as clearly as drawing the parse trees (shown by {... {...}} not okay, by adding extra { } you actually get rid of the ambiguity, and confuse your reader. Verbal explanations are always confusing. We've been telling you to show ambiguity by drawing two parse trees.)

+ 0 pts No valid explanation / not understanding ambiguity / claiming that there's no ambiguity

6.2 6b 9 / 10

✓ **+ 2 pts** **First part: Knowing that the problem is left/right association**

✓ **+ 2 pts** **First part: Showing that you have an idea how to fix this ambiguity in general**

✓ **+ 2 pts** **First part: a right method of solving ambiguity, such as (1) fixing the "left-combine" or "right-combine" order (introducing new nonterm), or (2) changed the defining the {statement} (expression) {statement} or other indication (actually we prefer (1), but the way the question is asked is kind of misleading so it is okay to be (2)) [Notice: changing the style back to if-then-else is NOT okay, it is changing the style and basic idea of Cnoif.]**

✓ **+ 2 pts** **First part: a correct answer with adequate explanation**

+ 2 pts Second part: knowing that C/C++ priority of execution could cause some issues (easily fixed by assigning the new statement highest priority, at least higher than while, etc.)

+ 1 pts (bonus) figure out that empty statements (like ; (a>b) ;) is also an issue here => disallow the empty statement

✓ **+ 1 pts** **(bonus) write down the corrected rule(s) explicitly**

+ 0 pts Having no idea about the answer

QUESTION 7

7 7 12 / 15

- 0 pts Correct

- 15 pts Did not attempt

- 5 pts no discussion regarding one of - JIT/interpreter/compiler

- 2.5 pts partially correct discussion for Java interpreter

- 2.5 pts partially correct discussion for JIT compiler

- 2.5 pts partially correct discussion for conventional compiler

✓ **- 3 pts** **no discussion of runtime memory requirements of binary/bytecode**

- 1.5 pts partial/incorrect discussion of memory requirements of binary/bytecode

- 2 pts no discussion of compiler code optimization/dynamic OO features

- 1.5 pts partial/incorrect discussion about request types/code structure

UCLA CS 131 Midterm, Winter 2020
100 minutes total, open book, open notes, closed computer
Write answers on exam.

Name: SRIRAM BALACHANDRAN Student ID: 805167463

1 (8 minutes). Define a "varying point" of a function f to be a point x such that f x != x. Write an OCaml function (computed_varying_point eq f x) that preferably returns (Some v) where v is a varying point for f, and that returns None if it is not possible to find a varying point for f from the information given. As with Homework 1's computed_fixed_point function, eq is the equality predicate for f's domain and x is a member of f's domain. Give the type of computed_varying_point, which should be as general as possible.

```
let computed_varying_point eq f x =
  match (eq x (f x)) with
    | true -> None          (* if fx = x, then recursing will yield some result *)
    | false -> Some x

;;
computed_varying_point : ('a -> 'a -> bool) -> ('a -> 'a) -> 'a?
```

2 (10 minutes). Consider the following function, taken from the hint code for Homework 2.

```
let match_nucleotide nt frag accept =
  match frag with
    | [] -> None
    | n::tail -> if n == nt then accept tail else None
```

This function matches a single nucleotide 'nt'. Write a similar function match_run that has match_nucleotide's API, but that instead matches a sequence of one or more nucleotides equal to 'nt', followed either by the end of the fragment or by a nucleotide not equal to 'nt'.

```
let match_run nt frag accept = function
    | [] -> accept []
    | h::t -> if h=nt then (match_run nt t accept) else accept frag
;;
```

3. The java.util.Collections class defines a method with this API: Generic

```
public static <T> List<T>
  unmodifiableList(List<? extends T> list);
```

Return an unmodifiable view of the specified list. That is, if v = unmodifiableList(u), then v is like u except that you cannot change the list by acting on v; you must change it by acting on u instead.

3a (10 minutes). Suppose ls is of type List<String> and lo is of type List<Object>. Explain why the assignment 'lo = ls;' is invalid, but the assignment 'lo = Collections.unmodifiableList(ls);' is OK (even if ls changes after lo is assigned to).

Object Object

The assignment 'lo = ls;' is invalid because this assumes that List<String> is a subtype of List<Object>, ~~this however, does not make~~ that is not true. This can be illustrated with a simple example. Say 'lo = ls;' was valid. Now lo contains a reference to a List<String> object. Then say I tried to do ~~I add~~ 'lo.add(new Thread())'. This ~~was~~ should be allowed, since lo ~~should~~ is of type 'List<Object>'. However, lo contains a reference to a 'List<String>', and you can't put a thread in a list of Strings. However, 'lo = Collection unmodifiableList(ls);' is ok since it takes in an argument of List<? extends T>. In the assignment, it is inferred that T in this case is 'Object', and if we take ? = String, then String extends Object which is true, and we have fulfilled the contract, returning a List<Object> type.

3b (10 minutes). Given the above, explain why the type T of the expression unmodifiableList(X) cannot in general be inferred merely by looking at the type of X. What else can affect T? Explain with a brief example.

T can be affected by the expression in which Collections.unmodifiableList is used in. In the same way that 'long x = 5;' forces us to treat 5 as a long, 'lo = Collections.unmodifiableList(ls);' forces us to consider T = object by virtue of the fact the LHS of the assignment is of type List<Object>. Also, because of the fact the returned reference is unmodifiable itself, there's no danger of the 'lo.add(new Thread());' example mentioned above. As a result, it is okay for the return type T to be a superclass of ?, which necesitates the need for the <? extends T> in the arguements

[page 4]

4. For each of the following OCaml expressions, give its type and value.

4a (3 minutes). `let rah f g = g f`

rah : 'a -> ('a -> 'b) -> 'b

4b (3 minutes).

```
let expr = "3 + 4"
in let lvalue = "a"
   in [expr, ["("; expr; ")"];
       lvalue, ["$"; expr]]
```

(string * (string list)) list

4c (6 minutes).

```
let moo a n = function
    | [] -> None
    | h::t -> if h == n then a h t else None
```

moo : 'a -> 'a -> 'b?) -> 'a -> 'a list -> 'b?

5. Suppose you want a wrapper class that contains a value of type T and has a getter 'get' and a setter 'set' to load and store a value. The getter returns null until the setter is called with a non-null value, and and later calls to 'set' are no-ops. You want the wrapper to be safe and fast in a multithreaded application. Consider the following alternative implementations of the wrapper class:

```
class C0<T> {
  T val;
  public T get() { return val; }
  public void
  set(T v) {
    if (val == null) { val = v; }}}
```

```
class C1<T> {
  volatile T val;
  public T
  get() { return val; }
  public synchronized void
  set(T v) {
    if (val == null) { val = v; }}}
```

```
class C2<T> {
  T val;
  public T get() { return val; }
  public synchronized void
  set(T v) {
    if (val == null) { val = v; }}}
```

```
class C3<T> {
  T val;
  public synchronized T
  get() { return val; }
  public synchronized void
  set(T v) {
    if (val == null) { val = v; }}}
```

5a (12 minutes). Which of the classes are safe in a multithreaded application? Briefly justify your answers by appealing to the JMM.

C3 is the best multithreaded-safe implementation, Although C1 seems appealing due to T being a volatile member variable, this does not solve the issue in question — it merely forces both threads to directly access & modify the T value stored in RAM, but doesn't prevent race conditions with 1 thread trying to do a get() while another one calls set(T v). C2 also falls short for the same reason — get and set access the same variable, and thus should not execute simultaneously on multiple threads, but C2's implementation allows this to happen. C3 is the best — both set & get have been synchronized with 'this' as the monitor, thereby ensuring only one thread modifies or accesses the T val at once.

5b (8 minutes). Explain the performance implications of the classes. Order them roughly in order of performance. State your assumptions.

C0 > C2 > C1 > C3. This is under the assumption the instance of the wrapper class has been initialized before entering thread work.

C0: no waiting ever, each thread is blind to other threads

C2: Assuming volatile access is much slower than since volatile accesses can't be optimized by JMM, C2 will be faster than C1.

C1: See above

C3: This will be slowest since both methods on the class are synchronized. This effectively means no 2 threads can ever call methods on the same instance at once, whereas the other implementations do allow some concurrency in the calls.

[page 6]

6. Suppose I design a new language Cnoif that is like C except it solves the dangling-if problem in a different way. In Cnoif, 'if' and 'else' are not reserved words, and every C 'if' statement is written like this:

```
statement (expression) statement
```

That is, there is no elseless 'if' statement: all 'if' statements have a 'then' statement that precedes the 'if' test, and an 'else' statement that follows the 'if' test. For example, this C statement:

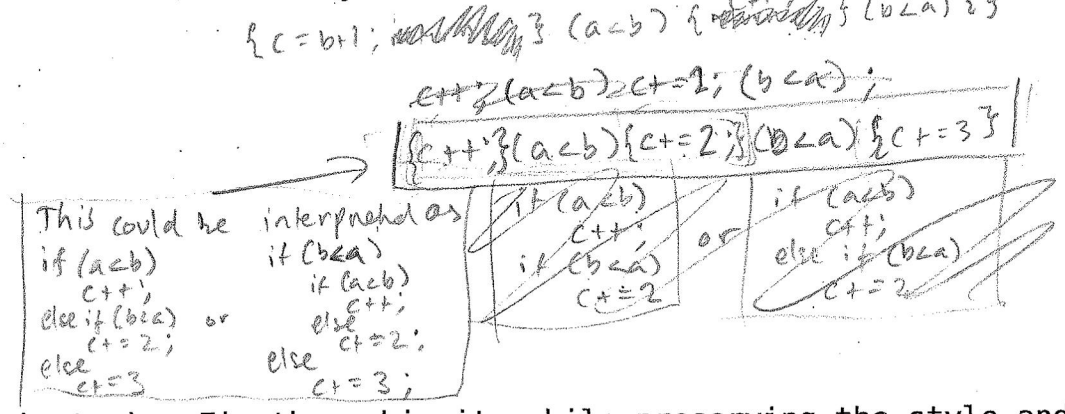
```
if (a < b) { c = b + 1; return c * c; }
```

can be written this way in Cnoif:

```
{ c = b + 1; return c * c; } (a < b) {}
```

Unfortunately the syntax of Cnoif is ambiguous.

6a (5 minutes). Illustrate the ambiguity with a program that can be parsed in two different ways.



6b (10 minutes). Fix the ambiguity while preserving the style and basic idea of Cnoif.

The ambiguity can be fixed simply by requiring that the 'statements' preceding the (expression) be marked by some ~~characters~~ terminals i.e: '{}' so the revised if rule looks like this:

{statement} (expression) {statement}

Define a nonterminal noifstatement, that allows every statement except an if. if statement can then be written as the combination of the following rules

statement (expression) noifstatement

or

statement (expression) {statement}

[page 7]
7 (15 minutes). You are optimizing a large Java server app to run on lnxsrv10.seas.ucla.edu, an Intel Xeon Silver 4116 which has 12 cores, each with (dual) hyperthreading, 32 KiB 8-way L1 I cache, a similar L1 D cache, and a 1 MiB L2 cache; and which has 16.5 MiB shared L2 cache and 64 GiB DDR4-2400 DRAM. All caches are write-back. This app will accept many varied requests from the network, and send responses via the network. The app does little or no floating point arithmetic, accesses a large number of small, shared Java objects, and has a lot of Java classes and code. You are mainly worried about average latency, that is, the average amount of time between a request and the corresponding response.

optimizations can be done, multithreading

Discuss the pros and cons of using (1) a plain Java interpreter, (2) a just-in-time Java compiler, and (3) a conventional compiler like gcj (which is GCC for Java), in the light of this application. Show why any of the three possibilities might be best for average latency, depending on the application's characteristics.

1. The app is large + and compiling it may take large amounts of time, which is not ideal if there is a quick bug fix that needs to happen b/c the server could be down for a while. Interpreting avoids this at the cost of speed and safety. ~~However if you only have a few common requests either~~ If you have a server application that needs to be updated frequently, ~~interpreting may be ideal~~ that doesn't ~~have to be~~ frequently use a large variety of objects, choice 1 may be ideal.

2. JIT could be ideal if your application has a large number of requests that don't frequently use a large variety of classes. This gives us the speed and safety of compilation with the flexibility of interpreting. This also allows the JMM to make multithreading-safe optimizations which would be useful for a server app.

3 Compilation would be most useful if you have a variety of complex, multi-object spanning calculations invoked by requests frequently (i.e. this is ideal for complex frequent requests)