

COM SCI 131 Midterm

Junhong Wang

TOTAL POINTS

87 / 100

QUESTION 1

11 7 / 8

+ 0 pts Incorrect

+ 7 Point adjustment

QUESTION 2

2 2 5 / 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 3 / 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 6 / 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 in severe 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 9 / 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

(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 6

6 15 pts

6.1 6a 5 / 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

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

Name: Junhang Wang Student ID: 50494113

1 (8 minutes). Define a "varying point" of a function f to be a point x such that $f x \neq x$. 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 rec computed_varying_point eq f x =
  if not(eq (f x) x) then x
  else computed_varying_point eq f (f x)
```

$$((('a \rightarrow 'a) \rightarrow \text{bool}) \rightarrow ('a \rightarrow 'a) \rightarrow 'a) \rightarrow 'a$$

[page 2]

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 = ...  
  let match_empty frag accept = accept frag in  
  let append_matchers matcher1 matcher2 frag accept =  
    matcher1 frag (fun s -> matcher2 s accept)  
  in  
  let rec match_run_helper = function  
    | [] -> match_empty  
    | head :: tail -> append_matchers (match_nucleotide head) (match_run_helper tail)  
  in  
  match_run_helper nt
```


[page 3]

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

```
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).

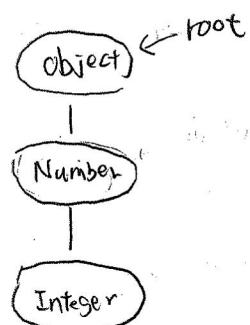
Consider the code snippet below:

```
1 List<String> ls = new List<String>();  
2 List<Object> lo = ls;  
3 lo.add(new Object());  
4 String s = ls.get(0); // problem!
```

At line 4, we're returning `Object` type because we inserted it at line 3. However, `ls` is a `String` list, so it expects the `get()` to return a `String`. Thus, it is invalid.
(Note: `String` is subclass of `Object`)

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.

Consider the class tree below:



→ However, if at line 2, we do
"lo = Collections.unmodifiableList(ls);"
then it won't allow line 3 because it's modifying the list. Thus, we won't have the issue we had before, and thus it is valid.

Note `Number` extends `Object` and `Integer` extends `Number`, which also extends `Object`. Thus, when we say `unmodifiableList(new List<Number>())`, the type `T` could be `Number`, but it also could be `Object`. Thus, we can't infer `T` by just looking at `X` (i.e. `Number`).

[page 4]

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

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

type: $('a \rightarrow ('a \rightarrow 'b)) \rightarrow 'b$

value: a function "rah", which calls second argument g with
first argument f as the argument to g, and returns whatever the function
g returns.

4b (3 minutes).

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

type: (string * string list) list

value: ["3+4", ["("; "3+4"; ")"];
"a", ["\$"; "3+4"]]

4c (6 minutes).

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

type: $((('a \rightarrow 'a list) \rightarrow 'b option) \rightarrow 'a \rightarrow 'a list) \rightarrow 'b option$

value: a function "moo", which takes three arguments:

"a" of type $('a \rightarrow 'a list) \rightarrow 'b option$,
"n" of type ' a ',
a list of type ' a list'.

[page 5]

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 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; }}}
```

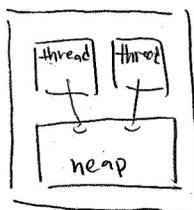
```
class C1<T> {  
    volatile T val;  
    public T  
    get() { return val; }  
    public synchronized void  
    set(T v) {  
        if (val == null) { val = v; }}}
```

```
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.

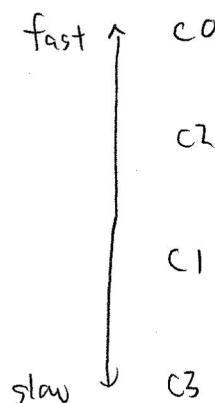
JMM (Java Memory Model) describes how multi-threaded applications are handled



Basically, all objects are created in the heap, and threads share the heap and can have pointers to objects in the heap.

C0 suffers from race conditions because multiple threads could modify "val" at the same time.

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



C0 is fastest because there is no synchronization at all.

C2 is pretty fast because there is only one "synchronized" keyword used.

"volatile" keyword ensures that the variable with the keyword is never cached. Thus, its change is visible to all threads. I'll assume synchronized get() is slower than volatile val. (This is reasonable assumption if there are many threads calling get())

Then C1 is faster than C3.

C1 won't have problem C0 had because there is "synchronized" keyword on set() function, which ensures that at most one thread can call set() and modify "val"

C2 and C3 won't have problems for the same reason as C1.

Therefore, C1, C2, and C3 are safe.

[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 $c * c$; }

$\{ \} \{ \} () \{ \} \{ \}$

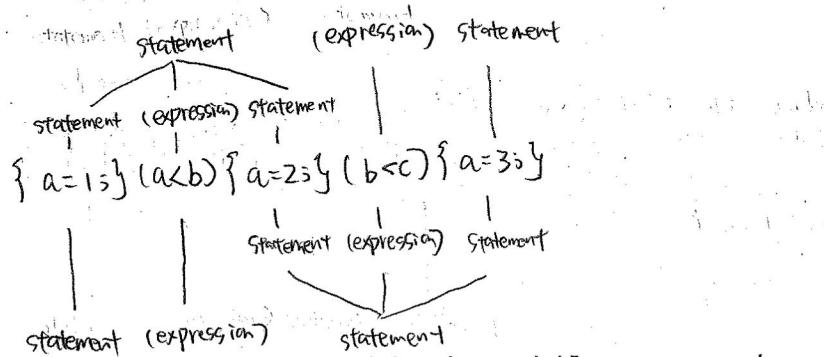
can be written this way in Cnoif:

{ $c = b + 1$; 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.

Assumption: Asked Kimmo during exam and he said {} is optional (i.e. " $a=1; (a < b) a=2;$ " is allowed based on the grammar above)



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

The problem with 6a is that we don't know when the if statement starts, and when the else statement ends.

thus we can enforce {} before and after expression.

Then we know where the if statement is, and where the else statement is. we can fix it like below:

{ statement } (expression) { statement }

Then the example in 6a is no longer allowed.

It must be either

{ { a=1; } (a < b) { a=2; } } (b < c) { a=3; }

or
{ a=1; } (a < b) { { a=2; } (b < c) { a=3; } }

[page 7]

7 (15 minutes). You are optimizing a large Java server app to run on lnxsr10.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.

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.

Description

- (1) a plain Java interpreter just read the bytecode and execute them
- (2) a just-in-time Java compiler looks at the statistics of the bytecode executed, then it converts the frequently executed bytecode to machine code. Next time the bytecode is encountered, then it will be redirected to the machine code we generated, and run that machine code instead of the original byte code:
- (3) a conventional compiler just run the machine code.

Pros & cons

- (1) It's portable. If it works on one machine, then it must work in different machine that supports JVM (Java Virtual Machine), but it is not as fast as executing machine code, since it is executing bytecode.
- (2) This solves the problem of being slower than compiler by using machine code for frequently executed bytecode. The downside is that there is some overhead while converting the bytecode to machine code.
- (3) It's very fast/refficient because it's just executing machine code. But it is not portable. Depending on how the compiler is written, we may observe different behavior of the program on different machine.

Analysis

Since the program is written in Java, and lnxsr10 supports JVM, we can do either (1) or (2). Because we want to minimize the average latency (not longest latency), so I would pick (2) since it may slow down sometime but overall faster than (1).

