ECE 467 Final
Exam Type: C

University of Toronto
Faculty of Applied Science and Engineering
Examiner: Adrian Chiu

2022 December 19
2.5 Hours

Question	Points	Score
1	7	
2	3	
3	4	
4	6	
5	6	
6	6	
7	5	
Total:	37	

1. Consider the following language L of even-length, 2-bit palindromes (2-bit meaning each character is one of two values, i.e. a bit).

   - The alphabet $\Sigma = \{0, 1\}$.
   - The empty string ϵ is in L.
   - If $s \in L$, then $0s0 \in L$.
   - If $s \in L$, then $1s1 \in L$.

   (a) (1 point) Treating the alphabet as the set of terminals, write a context-free grammar that defines this language.

   (b) (1 point) 1001 is a string in L. Verify this by writing a derivation for it with respect to your grammar.

   (c) (1 point) Your grammar is not LR(1). How *would* you show that it is not LR(1) (you do not actually have to carry out the process).

(d) In general, a grammar might not be LR(1), but the language it accepts may be accepted by another grammar which is LR(1). We wish to determine that *no possible* LR(1) grammar defines the language L.

Suppose we had an LR(1) grammar and corresponding LR(1) parsing table that accepted L (upon applying the LR(1) parsing algorithm on input strings from L). Refer to this as the parser P, e.g. "P accepts (strings from) L".

Since the string $1001 \in L$, P accepts 1001. In particular:

- From the start state, P has valid actions for each next character/terminal of 1001.
- Upon reaching the end of the input 1001 (i.e. given \$ as the lookahead), it accepts, i.e. *it reduces by the augmented start rule.*

i. (1 point) Consider the execution of P on 1001, and 00 which is also in L. In each case, upon reaching the end of input (i.e. given \$ as the lookahead), what state is P in?

ii. (2 points) Consider the execution of P on $10011001 \in L$, and $001001 \notin L$.
Assuming P accepts all strings in L, why does P also accept 001001?

- -0.5 for incorrect or unclear reasoning.
- 0.5 if left blank.

(e) (1 point) Can we define L using a regular expression? If so, give an example. If not, why?

Total for Question 1: 7

2. Consider the following code.

```
x0 = 5;
while (x1 = phi(x0, x4); x1 != 1) {
    if (x1 % 2) == 0 {
        x2 = x1 / 2;
    } else {
        t = 3 * x1;
        x3 = t + 1;
    }
    x4 = phi(x2, x3);
}
return x1;
```

(a) (1 point) Show the values of all variables/names (when applicable) in the first iteration of the loop.

(b) (1 point) Show the values of all variables/names (when applicable) in the second iteration of the loop.

(c) (1 point) Does the loop terminate?

Total for Question 2: 3

3. Consider the data-flow analysis of constant propagation for three 32-bit (signed) integer variables x, y, z.

The lattice V for a single variable consists of $2^{32}+2$ values: $\top$, $\bot$, and each of the possible 32-bit (signed) integer values. A value in the domain of this constant propagation is a 3-tuple in the product lattice $V \times V \times V$.

The meet operator $\wedge$ for a single lattice V is defined as follows (where c, d are any two distinct 32-bit (signed) integer values).

x_1	x_2	$x_1 \wedge x_2$
$\bot$	$\bot$	$\bot$
$\bot$	d	$\bot$
$\bot$	$\top$	$\bot$
c	$\bot$	$\bot$
c	d	$\bot$
c	c	c
c	$\top$	c
$\top$	$\bot$	$\bot$
$\top$	d	d
$\top$	$\top$	$\top$

The meet operator for the product lattice is defined in terms of the individual lattices as follows.

$$(x_1, y_1, z_1) \wedge (x_2, y_2, z_2) = (x_1 \wedge x_2, y_1 \wedge y_2, z_1 \wedge z_2).$$

The transfer function f_s for a statement s of the form $x = y + z$ is defined as follows, where $\_$ denotes any value (c, d are any 32-bit (signed) integer values).

(x, y, z)	$f_s(x, y, z)$
$(\_, \bot, \bot)$	$(\bot, \bot, \bot)$
$(\_, \bot, d)$	$(\bot, \bot, d)$
$(\_, \bot, \top)$	$(\top, \bot, \top)$
$(\_, c, \bot)$	$(\bot, c, \bot)$
$(\_, c, d)$	$(c + d, c, d)$
$(\_, c, \top)$	$(\top, c, \top)$
$(\_, \top, \bot)$	$(\top, \top, \bot)$
$(\_, \top, d)$	$(\top, \top, d)$
$(\_, \top, \top)$	$(\top, \top, \top)$

(a) (1 point) Write an equivalent definition for following inequality using the meet operator.

$$(x_1, y_1, z_1) \leq (x_2, y_2, z_2).$$

(b) (2 points) Select values (a, b, c) and (d, e, f) such that the following is true.

$$f((a, b, c) \wedge (d, e, f)) \lneq f(a, b, c) \wedge f(d, e, f).$$

- -1 point for incorrect computation or result.
- 0.5 for leaving this question blank.
- Note: this question is *not* marked with respect to your previous definition. If you are not confident in your previous definition, you *may* wish to leave this question blank.

(c) (1 point) How do the "meet over paths" (MOP) solutions compare to the "maximal fixed point" (MFP) solutions for this data-flow analysis?

Total for Question 3: 4

4. Relaxed atomic memory ordering is often only sufficient for counters. However, in general, it is *not* sufficient for thread-safe reference counting. Consider the following code.

```
struct RefCountedData {
    std::atomic<int> count;
    Data data;
};

void increment(RefCountedData* rc) {
    int old_count = rc->count.load(relaxed);
    rc->count.store(old_count + 1, relaxed);
}

void decrement(RefCountedData* rc) {
    int old_count = rc->count.load(relaxed);
    rc->count.store(old_count - 1, relaxed);
    if (old_count == 1) {
        // Destroy data
        rc->data.~Data();
    }
}
```

(a) (3 points) Consider the `decrement` operation. Describe how it may be possible for the thread destroying `data` to read stale memory for `data`.

CLARIFICATION: there are other potential problems with the code. Specifically, this question is concerned with the access of the `data` field in the `decrement` operation.

Hint: consider multiple threads using `data`*, and then calling* `decrement`*.*

- -1 point for incorrect or unclear reasoning.
- 1 point if left blank.

(b) (1 ½ points) What is the problem if *only* the `load` is changed to an acquire ordering?

- -0.5 for incorrect or unclear reasoning.
- 0.5 if left blank.

(c) (1 ½ points) What is the problem if *only* the `store` is changed to a release ordering?

- -0.5 for incorrect or unclear reasoning.
- 0.5 if left blank.

Total for Question 4: 6

5. Suppose we have the following pseudocode implementation of a stack-based interpreter that operates only on integers.

```
while (bytecode != nullptr) {
    switch (*bytecode) {
        case "load x":
        push(local_variables[x]);
        bytecode++;

        case "store x":
        local_variables[x] = pop();
        bytecode++;

        case "constant c":
        push(c);
        bytecode++;

        // CORRECTION: added "less" operation
        case "add/sub/mul/div/less":
        right = pop();
        left = pop();
        push(left add/sub/mul/div/less right);
        bytecode++;

        case "jump n":
        bytecode += n;

        case "test n":
        condition = pop();
        if (condition != 0) {
            bytecode++;
        } else {
            bytecode += n;
        }
    }
}
```

Consider the following C code.

```
t = 1;
for (int i = 0; i < 10; i++) {
    t = t * 2 + 1;
}
```

(a) (2 points) Draw a control-flow graph for the above C code (break up the clauses of the for-loop). You may group multiple operations into basic blocks.

- -0.5 for each mistake.

(b) (2 points) Translate the code in *each individual node* into bytecodes for the interpreter above. For `jump` and `test` bytecodes, you do not need to include the offset.

- -0.5 for each mistake.

(c) (2 points) Combine the fragments of bytecodes from each node into *a single array of bytecodes.* Then, fill in the offsets for `jump` and `test`.

- -0.5 for each mistake.

Total for Question 5: 6

6. Suppose we have *some* GC algorithm that satisfies the tri-color invariant:

   - All objects are labeled exactly one of: black, grey, or white.
   - No black-labeled object contains a pointer to a white-labeled object.

   You are given no other information about this GC algorithm; reason about it based on transitions between colours.

   (a) (1 point) What is a terminating condition for this GC algorithm?

   (b) (3 points) What are all the transitions we should disallow to ensure that the GC algorithm will terminate?

   - -1 point for incorrect or missing answer.
   - 1 point if left blank.

   (c) (2 points) Suppose this GC algorithm may run concurrently with the mutator threads (by the general definition of concurrent). The mutators may allocate objects while GC is running. What colour should newly allocated objects be labeled to ensure that the algorithm terminates, and why?

   - -1 point for incorrect or unclear reasoning.
   - 0.5 points if left blank.

Total for Question 6: 6

7. Based on the lab.

   (a) (1 point) Draw the general control-flow graph/basic block structure for a while loop *without* `break` or `continue` statements in the body.

   (b) (2 points) Ignoring nested loops, how would you support `break` and `continue`?

   - No penalty for error.

   (c) (2 points) Suppose `break` and `continue` only ever apply to the immediate enclosing loop. How would you extend your solution in the previous part to handle nested loops? For example, the following is valid code.

```
while (...) {
    while (...) {
        if (...) {
            break_again = true;
            break;
        }
    }
    if (break_again) {
        break;
    }
}
```

   - No penalty for error.

   (d) (0 points) Have a good holiday break.

Total for Question 7: 5