24wi Final Review

These practice questions are from CSE 331 2024 Winter. Questions relating to content that was not covered in 2025 Summer were removed. The remaining questions are similar to the style of questions you can expect to see on the 25SU exam and cover topics that are fair game, but it is not a perfect representation of what topics will be emphasized in the 25SU exam or the length.

Question 1 : Testing Subdomains

a. For the following function, select one input that is **not** required to test using the 331 Testing Rules (assuming all other options would be tested)?

```
const foo = (x: number) : number => {
  if (x < 0) {
    const a = x * 2;
    const b = a - 2;
    return b * 3 + 1;
  } else if (x > 10) {
    return 100 * x  - x / 2
  } else {
    return 55;
  }
}
```

A. 0
B. 13
C. 1
D. -1

b. For the following function, what is the **minimum** number of tests needed?

```
const bar = (x: number) : number => {
  if (x === 0) {
    return 0;
  } else {
    if (x % 2 === 0) {        // x is even
      return bar(x / 2);
    } else {                  // x is odd
      return bar(x - 1);
    }
  }
}
```

A. 1
B. 2
C. 3
D. 4
E. 5
F. 6
G. 7
H. 8+

Question 2: Types

Pick the best, most precise definition for what this inductive data type represents.

$$\text{type T} = \text{A(x: } \mathbb{Z}\text{)} \mid \text{B(x: } \mathbb{Z}\text{, y: T, z: T)}$$

A. Lists of integers
B. Non-empty lists of integers
C. Binary trees of integers
D. Non-empty binary trees of integers

Question 3: Proof By Calculation

For each of the proofs below, select the correct option to fill in the blank (____) space in order for the proof to be deemed correct.

Prove that $\text{sum}(\text{twice}(L)) = 2\text{sum}(L)$ where $L = \text{cons}(a, \text{cons}(b, \text{nil}))$

$\text{sum}(\text{twice}(L)) = \text{sum}(\text{twice}(\text{cons}(a, \text{cons}(b, \text{nil}))))$	__ P1 __.
$= \text{sum}(\text{cons}(2a, \text{twice}(\text{cons}(b, \text{nil}))))$	def of twice
$= \text{sum}(\text{cons}(2a, \text{cons}(2b, \text{twice}(\text{nil}))))$	def of twice
$= \text{sum}(\text{cons}(2a, \text{cons}(2b, \_\_\text{P2}\_\_)))$	def of twice
$= 2a + \text{sum}(\text{cons}(2b, \text{nil}))$	def of sum
$= 2a + 2b + \text{sum}(\text{nil})$	def of sum
$= 2a + 2b + 0$	def of sum
$= 2(a + b + 0)$	algebra
$= 2(a + b + \_\_\text{P3}\_\_)$	def of sum
$= 2(a + \text{sum}(\text{cons}(b, \text{nil})))$	def of sum
$= 2(\text{sum}(\text{cons}(a, \text{cons}(b, \text{nil}))))$	def of sum
$= 2(\text{sum}(L))$	def of L

Part 1:

What should go in the blank *reasoning* space on the 1st line of the proof:

A. def of List
B. def of cons
C. def of L
D. def of sum

Part 2:

What should go in the blank space on the 4th line of the proof: $\text{sum}(\text{cons}(2a, \text{cons}(2b, \_\_\text{P2}\_\_)))$

A. $\text{twice}(0)$
B. 0
C. nil
D. $\text{twice}(\text{nil})$

Part 3:

What should go in the blank space on the 9th line of the proof: $2(a + b + \_\_\text{P3}\_\_)$

A. $\text{sum}(\text{nil})$
B. nil
C. 0
D. $\text{sum}(0)$

Question 5

For each of the programs below, determine whether there will be a type error using both structural (as in TypeScript) and nominal (as in Java) typing. Assume the following types and variables are in scope for all of the programs.

```
type A = {x: bigint}
type B = {x: bigint}
type C = {x: number}
type D = {x: bigint, y: bigint}
type E = {y: bigint, x: bigint}

const a : A = {x: 123n}
const b : B = {x: 456n}
const c : C = {x: 789}
const d : D = {x: 1n, y: 2n}
const e : E = {y: 3n, x: 4n}
```

Program	Structural Typing Error (Yes/No)	Nominal Typing Error (Yes/No)
`v0: E = {x: 12n, y: 34n}`		
`v1 : B = a`		
`v2 : C = b`		
`v3 : D = e` `v4 : E = d`		
`v5 : A = e`		
`v6 : E = a`		

`foo = (x: E) : D => {`
`  return {x: 123n, y: 456n}`
`}`
`v7 : D = foo(e)`		

`foo = (x : A) : B => {`
`  return (x.x > 0n) ? {x: 1n} : {x: -1n}`
`}`
`bar = (x : B) : C => {`
`  return (x.x === 0n) ? {x: -1.5} : {x: 2 ** 32}`
`}`
`v8 : C = bar(foo({x: 0n}))`		

`v9 : A = {x: null}`		
`v10 : B = {x : 0n}` `v11 : A = {x : v10.x}`		

Question 6: Equality

For each of the expressions below, write the boolean value (T/F) that the expression will evaluate to.

Code	Eval (T/F)
3n === 3	
“I love 331” === “I love 331”	
5n == 5	
10.0 === 10	
false === 5 < 4	

`const a: {x: bigint, s: string} = {x: 4n, s: “hello”};`
`const b: {x: bigint, s: string} = {x: 5n, s: “hello”};`
 	
a === b	

`const a: {x: bigint, s: string} = {x: 5n, s: “hello”};`
`const b: {x: bigint, s: string} = {x: 5n, s: “hello”};`
 	
a === b	

`const a : {b: boolean, s: string,  x: bigint}`
`  = {b: true,  s: “okapi”, x: 1n};`
 	
`const b : {a: bigint, b: string,  c: boolean}`
`  = {a: 1n,  b: “giraffe”, c: false};`
 	
b.a === a.x	

“101” === 101	
“horses” === “horse” + “s”	

`const x = 5/2;`
`const y = 2;`
 	
x === y	

undefined === null	

Question 8: Inductive Data Types in TypeScript

For each of the following, select True if the TypeScript type is a correct encoding of the mathematical inductive data definition.

Math Definition	TypeScript Encoding	Match? T/F

type $X = A \mid B \mid C(x: X)$	`type X =`
	`  {kind: “A”}`
	`| {kind: “B”}`
	`| {kind: “C”}`	

type $Y = A(n: \mathbb{Z}) \mid B(n: \mathbb{Z})$	`type Y =`
	`  A(n: bigint)`
	`| B(n: bigint)`	

type $Z = A \mid B(z : Z)$	`type Z =`
	`  {kind: “A”}`
	`| {kind: “B”, z: Z}`	

type $U =$
$A(x: \mathbb{Z}) \mid B(x: \mathbb{Z}, y: U, z: U)$
	`type U =`
	`  {kind: “A”, x: bigint}`
	`| {kind: “B”, x: bigint, y: U, z: U }`	

type $V = A(r: \mathbb{R}) \mid B(n: \mathbb{Z})$	`type V =`
	`  {r: number}`
	`| {n: integer}`	

Question 9: Implications

For each of the following, answer T or F based on if the facts imply the obligation.

Facts	Obligation	Do the facts imply the obligation? T/F
For $x, y, z : \mathbb{N}$, $x = y$ and $2y > z$	$x > z/3$	
For $x, y: \mathbb{Z}$, $x \geq 9$ and $y \geq -10$	`return x + y` returns a positive integer	
For $x, y: \mathbb{Z}$, $x + y \leq -5$ and $y = 2$	$-x \leq 7$	
For $x, y, z : \mathbb{Z}$, $x = z - y$ and $y \geq 0$	$x \leq z$	

Question 10: Structural Induction

The following proof is incorrect. In one sentence, identify an error in the proof. In another single sentence, explain how to fix the proof either by adding, changing, or removing steps.

func concat(nil, R) := R for any R : List

concat(cons(x, L), R) := cons(x, concat(L, R)) for any x : $\mathbb{Z}$ and any L, R : List

Prove that len(concat(S, R)) = len(S) + len(R)
This proof is by structural induction on S.

Base Case: S = nil.
Goal is to show len(concat(nil, R)) = len(nil) + len(R)

len(concat(nil, R))	= len(R)	def of concat
	= 0 + len(R)	
	= len(nil) + len(R)	def of len

Inductive Hypothesis: assume that len(concat(L, R)) = len(L) + len(R)

Inductive Step S =(cons(x, L)).
Goal is to show len(concat(cons(x, L), R) = len(cons(x, L)) + len(R)

len(concat(cons(x, L), R))	= len(cons(x, concat(L, R)))	def of concat
	= 1 + len(L) + len(R)	Ind. Hyp.
	= len(cons(x, L)) + len(R)	def of len

Question 11: Induction Hypothesis

Recall the definition of Trees: $\text{type Tree} := \text{leaf} \mid \text{node}(\text{x} : \mathbb{Z}, \text{L} : \text{Tree}, \text{R} : \text{Tree})$
Suppose we are doing a proof of some property P by structural induction on S : Tree.
What induction hypothesis/es can we assume in the inductive case of the proof, where we have $\text{S} = \text{node}(\text{x}, \text{L}, \text{R})$?

A. P(S)
B. P(L)
C. P(R)
D. P(L) and P(R)
E. P(L) and P(S)
F. P(R) and P(S)

Question 12: Generic Types

Consider these types:

```
type A =
    | { kind: "M", value: A };

type B =
    | { kind: "P" }
    | { kind: "Q" };

type C =
    | { kind: "CA", value: bool }
    | { kind: "CB", value: B };

type D =
    | { kind: "DA", value: ℤ }
    | { kind: "DB", value: bool };

type E =
    | { kind: "EA" }
    | { kind: "EB", v1: [E, B] };

type F =
    | { kind: "FA", v1: F, v2: F] }
    | { kind: "FB" };
```

How many distinct values are there of each type (e.g., “zero”, “one”, “two”, ..., “infinity”)?

Question 13. Comparing Specifications

Here are four different specifications.

Spec A
@param x, a natural number
@returns a natural number between 1 and 10

Spec B
@param x, a natural number
@requires x is even
@returns a natural number between 1 and 10

Spec C
@param x, an integer between 0 and 10
@returns an integer between 0 and 20

Spec D
@param x, a natural number
@returns an integer between -10 and 10

For each pair of specs, fill in the blank if the first spec is stronger than, weaker than, or incomparable to the second.

A is _____ than/to B

A is _____ than/to C

B is _____ than/to C

B is _____ than/to D

C is _____ than/to D

D is _____ than/to A

Question 14. ADTs

Consider the following public specification for an ADT representing a Rectangle:

```
// A Rectangle is represented by a triple (p, l, w) where:
//   p is the location of the top-left corner, l is the length, and w is the width
export interface Rectangle {
  length: () => bigint;
  width: () => bigint;
  topLeft: () => {x: bigint, y: bigint};
}
```

For each of the following concrete representations of the Rectangle abstract data type, write down the abstraction function (AF) and representation invariant (RI), if any.

```
class TopLeftRectangle implements Rectangle {
  // AF: obj =
  // RI:
  readonly topLeft: {x: bigint, y: bigint};
  readonly length: bigint;
  readonly width: bigint;
}
```

```
class OppositeCornersRectangle implements Rectangle {
  // AF:
  // RI:
  readonly topLeft: {x: bigint, y: bigint};
  readonly bottomRight: {x: bigint, y: bigint};
}
```

```
class CenterRectangle implements Rectangle {
  // AF: obj =
  // RI:
  readonly center: {x: bigint, y: bigint};
  readonly length: bigint;
  readonly area: bigint;
}

Bonus: Give an example of a Rectangle you can represent in the other two
implementations but not this one
```

Question 15: Hoare Triples

For each of the following Hoare Triples, determine whether the triple is valid.

{{ $x < 0$ }} `y = 2n * x;` {{ $y \leq 0$ }}	
{{ $x \geq y$ }} `z = x - y;` {{ $z > 0$ }}	

{{ True }}
`if ( x >= 10n) {`
`  y = x % 7n;`
`} else {`
`  y = x - 1n;`
`}`
{{ $y < 9$ }}	

{{ $x < 0$ }}
`if (x < 100n) {`
`  x = -1n;`
`} else {`
`  x = 1n;`
`}`
{{ $x < 0$ }}	

`/**`
`* @param n: a natural number`
`* @returns a natural number`
`*/`
`const foo = (n: bigint): bigint => {`
`  …`
`}`
 	
// some other code that calls foo
{{ $x \geq 0 \wedge y \geq 0$ }}
`if (x < y) {`
`  y = -1n * x`
`}`
`z = foo(y)`
{{ $z \geq 0$ }}	

Question 17: Weakest Assertion

Select the weakest assertion in each set. The first row is filled in as an example.

Assertions	Weakest
A. {{ True }} B. {{ False }} C. {{ z = 42 }}	A
A. {{ x = 20 }} B. {{ x > 10 }} C. {{ x ≥ 10 }}	
A. {{ t = 2 }} B. {{ t ≠ 0 }} C. {{ t > 0 }}	
A. {{ x > 0 /\ y > 0 }} B. {{ x > 0 \/ y > 0 }}	
A. {{ |x + y| > w }} B. {{ x + y > w }}	

Question 18: Array Loop Invariants

```
/**
* Swaps two elements in an array
* @param A The array in which to swap elements
* @modifies A
* @param i The index of the first element
* @param j The index of the second element
* @requires 0 <= i < A.length /\ 0 <= j < A.length
*/
const swap = (A: bigint[], i: number, j: number): void => {
 const tmp = A[i];
 A[i] = A[j];
 A[j] = tmp;
}
/**
* Sorts an array in place
* @param A
* @modifies A
*/
const sort = (A: bigint[]): void => {
 let i = 0;
  // Inv: TODO
 while (i < A.length) {
   let j = i;
   // Inv: TODO
   while (j < A.length) {
     if (A[i] > A[j]) {
       swap(A, i, j);
     }
     j = j + 1;
   }
   i = i + 1;
 }
// Postcondition: A[x] ≤ A[y] for all 0 ≤ x < A.length and x ≤ y < A.length
```

What is the correct invariant for the outer loop?

A. $A[x] \leq A[y]$ for all $0 \leq x \leq i$ and $x \leq y \leq i$ and $0 \leq i < A.length$

B. $A[x] \leq A[y]$ for all $0 \leq x \leq i$ and $x \leq y \leq A.length$ and $0 \leq i < A.length$

C. $A[x] < A[y]$ for all $0 \leq x < i$ and $x \leq y < i$ and $0 \leq i < A.length$

D. $A[x] < A[y]$ for all $0 \leq x < i$ and $x \leq y < A.length$ and $0 \leq i < A.length$

E. $A[x] \leq A[y]$ for all $0 \leq x < i$ and $0 \leq i < A.length$ and $x \leq y < A.length$

What is the correct invariant for the inner loop?

A. $A[x] \leq A[j]$ for all $i < x < j$ and $0 \leq i < j < A.length$

B. $A[x] \leq A[j]$ for all $i \leq x \leq j$ and $0 \leq i < j < A.length$

C. $A[x] \leq A[y]$ for all $i \leq x < j$ and $x \leq y < j$ and $0 \leq i < j < A.length$

D. $A[x] \leq A[j]$ for all $i \leq x < A.length$ and $0 \leq i < j < A.length$

E. $A[i] \leq A[j]$ for all $0 \leq i < j < A.length$

Question 19: Servers and Routes

For each of the following, mark whether it applies to client/server interactions, normal function calls, both, or neither

	Client/Server Interaction	Normal Function Call
The public specification should be documented		
It is asynchronous		
The arguments must be serialized to JSON or text		
It can throw an error		
The function and the code that calls it are implemented in different languages		

Question 20: Stateful UI

```
type FooState = { items: string[] };

class Foo extends Component<{}, TodoState> {
...
// Called when the user clicks on the button to clear all items.
doClearClick = (_: MouseEvent<HTMLButtonElement>): void => {
       const name = this.state.input.trim();
       if (name.length === 0) {
              return;
       }

       // Now set the state to empty array
       ____________________________________
...
}
```

Which of the following options should go on the line in the method above?

A. this.state = {items: []};
B. this.setState({items: []});
C. Either A or B will work
D. Neither A nor B is correct.

Question 21: Status Codes

For each of the following scenarios, indicate the most appropriate status code for the response

Scenario	Error Code (200, 400, 500)
The request had all of the required parameters and the response contains the information the user was looking for about New Zealand	A. 200 B. 400 C. 500
The request is missing the country name (a required parameter), so the search cannot be completed	A. 200 B. 400 C. 500
The server receives the request and it has the required parameters, but there is a problem with the database, so the server is unable to complete the search	A. 200 B. 400 C. 500