Started on	Friday, 23 April 2021, 14:03
State	Finished
Completed on	Tuesday, 27 April 2021, 15:24
Time taken	4 days 1 hour
Grade	**40.00** out of 110.00 (**36**%)

Question 1

Incorrect Mark 0.00 out of 10.00

Consider the following implementation:

```
static int findFirst(int[] arr, int val) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == val) return i;
    }
    return arr.length;
}
```

and this specification of `find`:

```
static int find(int[] arr, int val)
  requires: nothing
  effects: returns largest index i such that
           arr[i] == val, or -1 if no such i
```

Which inputs demonstrates that `findFirst` does **not** satisfy this spec?

Select one or more:

- [ ] [1, 2, 2], 2
- [ ] [1, 2, 3], 2
- [ ] [1, 2, 3], 4
- [x] none of all others, `findFirst` does satisfy this spec!

Your answer is incorrect.

The correct answers are: [1, 2, 2], 2, [1, 2, 3], 4

Question 2

Incorrect Mark 0.00 out of 10.00

Consider the following implementation:

```
static int findLast(int[] arr, int val) {
    for (int i = arr.length-1 ; i >= 0; i--) {
```

```
        if (arr[i] == val) return i;
    }
    return -1;
}
```

and this specification of `find`:

```
static int find(int[] arr, int val)
  requires: nothing
  effects: returns largest index i such that
           arr[i] == val, or -1 if no such i
```

Which input demonstrates that `findLast` does **not** satisfy this spec?

Select one:

- a. [1, 2, 2], 2
- b. [1, 2, 3], 2
- c. [1, 2, 3], 4
- d. none of all others, `findLast` does satisfy this spec!

Your answer is incorrect.

The correct answer is: none of all others, `findLast` does satisfy this spec!

Question 3

Incorrect | Mark 0.00 out of 10.00

For each spec below, which one is **not** deterministic (underdetermined) ?

Select one:

a.
```
static int find(int[] arr, int val)
  requires: val occurs in arr
  effects:  returns index i such that arr[i] == val
```

b.
```
static int find(int[] arr, int val)
  requires: val occurs exactly once in arr
  effects:  returns index i such that arr[i] == val
```

c.
```
static int find(int[] arr, int val)
  requires: nothing
  effects: returns largest index i such that arr[i] == val, or -1 if no such i
```

d.
```
static int find(int[] arr, int val)
  requires: val occurs in arr
  effects: returns largest index i such that arr[i] == val
```

Your answer is incorrect.

The correct answer is:

```
static int find(int[] arr, int val)
  requires: val occurs in arr
  effects:  returns index i such that arr[i] == val
```

Question 4

Incorrect | Mark 0.00 out of 10.00

Given this specification:

```
static String join(String delimiter, String[] elements)
  effects: append together the strings in elements, but at each step,
           if there are more elements left, insert delimiter
```

Rewrite the spec so it is declarative, **not** operational.

Select one:

a.
```
effects: returns elements joined together with copies of delimiter, i.e.
         elements[0] + delimiter + elements[1] + delimiter +
         ... + delimiter + elements[elements.length-1]
```

b.
```
effects: returns the result of adding all elements to a
         new StringJoiner(delimiter)
```

c.
```
effects: returns the result of looping through elements and
         alternately appending an element and the delimiter
```

d.
```
effects: returns the result of recursive calls on the elements and
         while concatenating the delimiter
```

Your answer is incorrect.

The correct answer is:

```
effects: returns elements joined together with copies of delimiter, i.e.
         elements[0] + delimiter + elements[1] + delimiter +
         ... + delimiter + elements[elements.length-1]
```

Question 5

Incorrect | Mark 0.00 out of 10.00

When a specification is strengthened :

Select one:

a. fewer implementations satisfy it, and more clients can use it

b. fewer implementations satisfy it, and fewer clients can use it

c. more implementations satisfy it, and fewer clients can use it

d. more implementations satisfy it, and more clients can use it

Your answer is incorrect.

The correct answer is: fewer implementations satisfy it, and more clients can use it

Question 6

Correct | Mark 10.00 out of 10.00

Which of the following is **false** about a pair of specifications *A* and *B*?

Select one:

Select one:

- a. *A* can be stronger than *B* and have a stronger precondition
- b. *A* can be stronger than *B* and have a weaker precondition
- c. *A* can be stronger than *B* and have the same precondition
- d. *A* can be incomparable to *B*

Your answer is correct.

The correct answer is: *A* can be stronger than *B* and have a stronger precondition

Question 7

Incorrect | Mark 0.00 out of 10.00

Here are the `find` specifications from Lecture 8 :

```
static int find^ExactlyOne(int[] a, int val)
  requires: val occurs exactly once in a
  effects:  returns index i such that a[i] == val
```

```
static int find^OneOrMore,AnyIndex(int[] a, int val)
  requires: val occurs at least once in a
  effects:  returns index i such that a[i] == val
```

```
static int find^OneOrMore,FirstIndex(int[] a, int val)
  requires: val occurs at least once in a
  effects:  returns lowest index i such that a[i] == val
```

```
static int find^CanBeMissing(int[] a, int val)
  requires: nothing
  effects:  returns index i such that a[i] == val,
            or -1 if no such i
```

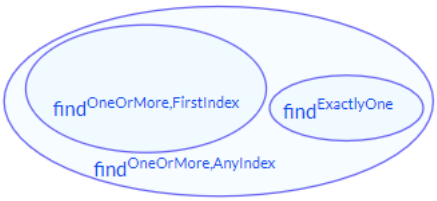
We already know that $\texttt{find}^{\texttt{OneOrMore,FirstIndex}}$ is stronger than $\texttt{find}^{\texttt{OneOrMore,AnyIndex}}$, which is stronger than $\texttt{find}^{\texttt{ExactlyOne}}$.

Where is $\texttt{find}^{\texttt{ExactlyOne}}$ on the diagram ?

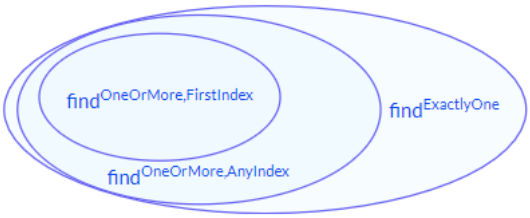
Select one:

a.

b.

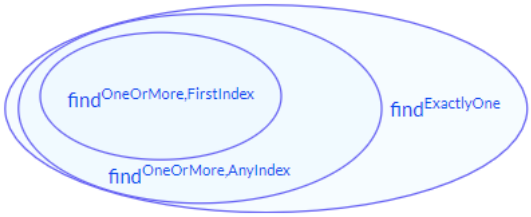


c.



d.

Your answer is incorrect.



The correct answer is:

Question 8

Incorrect Mark 0.00 out of 10.00

Here are the `find` specifications from Lecture 8 :

```
static int find^ExactlyOne(int[] a, int val)
  requires: val occurs exactly once in a
  effects:  returns index i such that a[i] == val
```

```
static int find^OneOrMore,AnyIndex(int[] a, int val)
  requires: val occurs at least once in a
  effects:  returns index i such that a[i] == val
```

```
static int find^OneOrMore,FirstIndex(int[] a, int val)
  requires: val occurs at least once in a
  effects:  returns lowest index i such that a[i] == val
```

```
static int find^CanBeMissing(int[] a, int val)
  requires: nothing
  effects:  returns index i such that a[i] == val,
            or -1 if no such i
```

We already know that $\texttt{find}^{\texttt{OneOrMore,FirstIndex}}$ is stronger than $\texttt{find}^{\texttt{OneOrMore,AnyIndex}}$, which is stronger than $\texttt{find}^{\texttt{ExactlyOne}}$.

How does $\texttt{find}^{\texttt{CanBeMissing}}$ compare to $\texttt{find}^{\texttt{ExactlyOne}}$?

Select one:

a. $\texttt{find}^{\texttt{CanBeMissing}}$ is stronger than $\texttt{find}^{\texttt{ExactlyOne}}$

b. $\texttt{find}^{\texttt{CanBeMissing}}$ is weaker than $\texttt{find}^{\texttt{ExactlyOne}}$

c. $\texttt{find}^{\texttt{CanBeMissing}}$ and $\texttt{find}^{\texttt{ExactlyOne}}$ are incomparable

d. none of the options is correct

Your answer is incorrect.

The correct answer is: $\texttt{find}^{\texttt{CanBeMissing}}$ is stronger than $\texttt{find}^{\texttt{ExactlyOne}}$

Question 9

Correct Mark 10.00 out of 10.00

Here are the `find` specifications from Lecture 8 :

```
static int find^ExactlyOne(int[] a, int val)
  requires: val occurs exactly once in a
  effects:  returns index i such that a[i] == val
```

```
static int find^OneOrMore,AnyIndex(int[] a, int val)
  requires: val occurs at least once in a
  effects:  returns index i such that a[i] == val
```

```
static int find^OneOrMore,FirstIndex(int[] a, int val)
  requires: val occurs at least once in a
  effects:  returns lowest index i such that a[i] == val
```

```
static int find^CanBeMissing(int[] a, int val)
  requires: nothing
  effects:  returns index i such that a[i] == val,
            or -1 if no such i
```

We already know that $\texttt{find}^{\texttt{OneOrMore,FirstIndex}}$ is stronger than $\texttt{find}^{\texttt{OneOrMore,AnyIndex}}$, which is stronger than $\texttt{find}^{\texttt{ExactlyOne}}$.

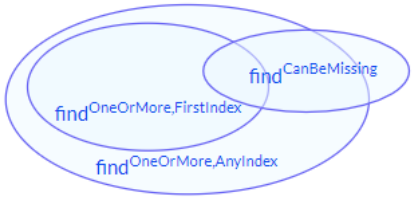
Where is $\texttt{find}^{\texttt{CanBeMissing}}$ on the diagram ?

Select one:

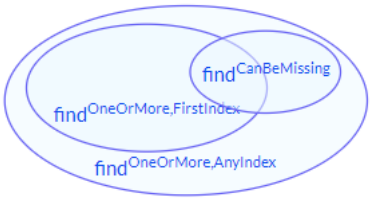
a.

b.

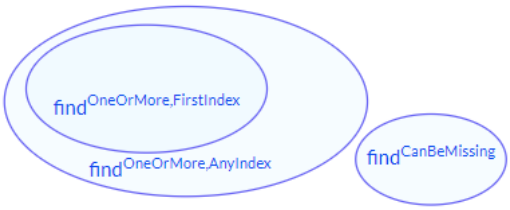
c.



d.



e.



Your answer is correct.

The correct answer is:

Question 10

Correct Mark 10.00 out of 10.00

In our ARList implementation, we use a technique called resizing array ✔ that doubles the size of the array whenever it is full.

Question 11

Correct Mark 10.00 out of 10.00

You want to use a generic array using casting in your implementation of a data structure.
For example, you write the following line in your constructor or your method:

```
T[] elements = (T[]) new Object[numOfElements];
```

Write the annotation that you need to write before the constructor or the method:

Answer: @SuppressWarnings("unchecked") ✔

The correct answer is: @SuppressWarnings("unchecked")

Finish review

Lab 8 Recording Jump to... Lab Exercise 8.1 ARDeque EMP