Debugging: Part 2

Practice Quiz, 11 questions

11/11 points (100%)

---

1. 1 / 1 points

You have fixed one bug in your friend's code, so that it no longer throws an index out of bounds exception, by adding the condition index >= input.length() - 3:

```
public void findAbc(String input){
    int index = input.indexOf("abc");
    while (true){
        if (index == -1 || index >= input.length() - 3){
            break;
        }
        String found = input.substring(index+1, index+4);
        System.out.println(found);
        index = input.indexOf("abc",index+3);
    }
}

public void test(){
    //findAbc("abccc");
    findAbc("abcdabc");
}
```

Let's test some more. Run the above code with input "abcdkfjsksioehgjfhsdjfhksdfhuwabcabcajfieowj". What is the output?

Enter your answer with the printed strings separated by commas, str1, str2. For example, if the output were "bcq", "bcd", and "bcu", you would write: bcq, bcd, bcu

bcd,bca

Correct Response

---

2. 1 / 1 points

Check this answer = do the problem by hand. What should be the correct output for input "abcdkfjsksioehgjfhsdjfhksdfhuwabcabcajfieowj"?

Enter your answer with the printed strings separated by commas, str1, str2. For example, if the output were "bcq", "bcd", and "bcu", you would write: bcq, bcd, bcu

bcd,bca,bca

Correct Response

---

3. 1 / 1 points

You will have to do some debugging, since the output wasn't what you were expecting. Let's see which occurrences of "abc" the program is finding. Add a line to print the index before **found** is calculated.

What are the indices printed? Select all that are correct.

- [x] 0

  Correct

- [ ] 1

  Un-selected is correct

- [x] 30

  Correct

- [ ] 31

  Un-selected is correct

- [ ] 33

  Un-selected is correct

---

4. 1 / 1 points

You can see that the program is finding the first two occurrences of "abc" but not the third. The while loop is breaking without finding this occurrence. So we know that when the variable **index** is updated after finding the second occurrence of "abc" at index 30, it must be updated either to -1 or to something greater than or equal to the length of **input** - 3. Let's see which it is.

Add a print statement. You might find it helpful to distinguish this from the print statement you added earlier so you can more easily see which is the index before updating and which is the index after. For example, you might do something like:

```
System.out.println("index " + index);
//code
System.out.println("index after updating " + index);
```

What is the value of index after updating for the last time?

-1

Correct Response

---

5. 1 / 1 points

Now we can tell that the code isn't finding the last occurrence of "abc" even though we can see that a third occurrence exists. At what index should the third occurrence of "abc" be found?

33

Correct Response

---

6. 1 / 1 points

After the program finds the 2nd occurrence of "abc", at what index does it start searching for the 3rd occurrence?

Hint: look at the line index = input.indexOf("abc",index+4);

34

Correct Response

---

7. 1 / 1 points

What are some other examples of input that would also have this problem? Select all that are correct.

- [x] "kdabcabcjei"

  Correct

- [ ] "tsabcesoeiabco"

  Un-selected is correct

- [ ] "abcbabccabcd"

  Un-selected is correct

- [ ] "qwertyabcuioabcp"

  Un-selected is correct

- [x] "abcabcabcabca"

  Correct

---

8. 1 / 1 points

Imagine your friend wants to get help from Coursera classmates on the discussion forums. Which of the following would be the most helpful way to describe the problem so that others can easily help?

- ( ) My method findAbc() isn't working.
- ( ) My method findAbc() is giving me the wrong answer.
- ( ) My method findAbc() works on "abcdabc" but is giving me the wrong answer with input "abcdkfjsksioehgjfhsdjfhksdfhuwabcabcajfieowj".
- (x) My method findAbc() works on "abcdabc" but is giving me the wrong answer with input "abcdkfjsksioehgjfhsdjfhksdfhuwabcabcajfieowj"; it prints "bcd, bca" when it should print "bcd, bca, bca". It also gives the wrong answer with input "kdabcabcjei", it prints "bca" when it should print "bca, bcj".

Correct

Always give your peers as much information about your problem as possible. However, posting your entire code is rarely useful.

---

9. 1 / 1 points

What is causing this bug?

- ( ) When the character following "abc" is "a" the program misses the next "abc"
- ( ) When one occurrence of "abc" is followed immediately by another occurrence of "abc", the while loop breaks
- (x) When one occurrence of "abc" is followed immediately by another occurrence of "abc", the method does not find the second "abc" because it starts searching at the "b" rather than at the "a" following the first "abc"

Correct

- ( ) The method will never find any occurrences of "abc" after the second one

---

10. 1 / 1 points

Which change needs to be made to fix the bug?

- ( ) Change the line

```
String found = input.substring(index+1, index+4);
```

to

```
String found = input.substring(index+1, index+3);
```

- ( ) Change the line

```
String found = input.substring(index+1, index+4);
```

to

```
String found = input.substring(index+1, index+5);
```

- (x) Change the line

```
index = input.indexOf("abc",index+4);
```

to

```
index = input.indexOf("abc",index+3);
```

Correct

Now the program will begin searching after the occurrence of "abc" rather than after the "bc_" following the "abc".

- ( ) Change line

```
index = input.indexOf("abc",index+4);
```

to

```
index = input.indexOf("abc",index+5);
```

---

11. 1 / 1 points

Make the change. Test your method on the input options that would have caused the bug (see question 7). What is the output when you run it with input "abcabcabcabca"?

Enter your answer with the printed strings separated by commas, str1, str2. For example, if the output were "bcq", "bcd", and "bcu", you would write: bcq, bcd, bcu

bca,bca,bca,bca

Correct Response
