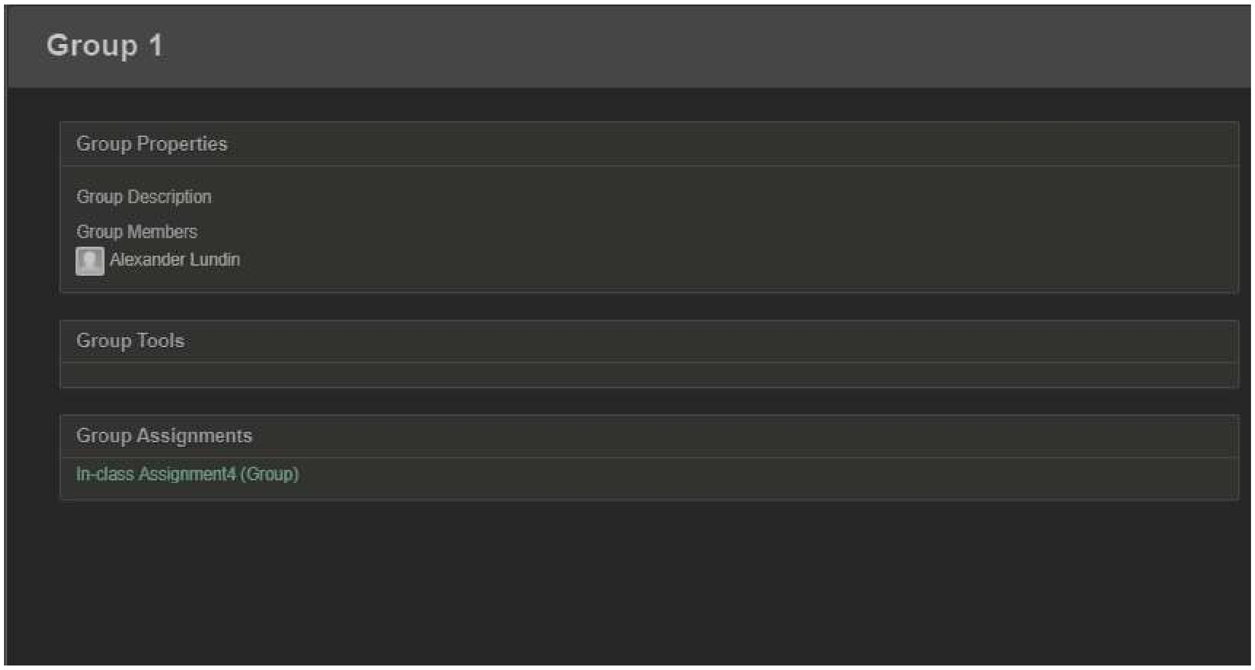
Alex Lundin
SE 4367.0U1

HW1

1.) Sign up for group discussion board



Group 1

Group Properties

Group Description

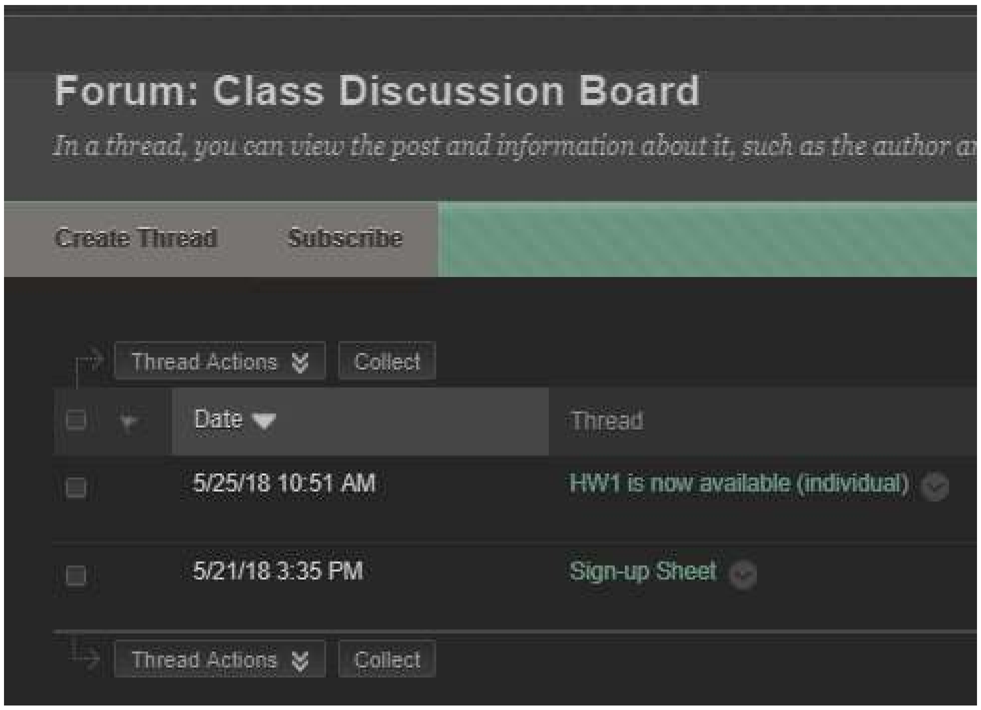
Group Members

Alexander Lundin

Group Tools

Group Assignments

In-class Assignment4 (Group)



Forum: Class Discussion Board

In a thread, you can view the post and information about it, such as the author a

Date	Thread
5/25/18 10:51 AM	HW1 is now available (individual)
5/21/18 3:35 PM	Sign-up Sheet

2.) Questions a-e apply to 4 pieces of code

a) Explain what is wrong with the given code. Describe fault precisely by proposing a modification to the code.
b) If possible, give a test case that DOES NOT execute the fault. If not, briefly explain why not.
c) If possible, give a test case that DOES execute fault, but DOES NOT result in an error state. If not, briefly explain why not.
d) If possible, give test case that results in error state, but DOES NOT result in a failure (hint program counters). If not, briefly explain why not.
e) For the given test case, describe the first error state. Be sure to describe it completely.

findLast

a) This piece of code will not be allowed to reach the very first index in the array. Changing the comparison part of the for loop to i >= 0 will fix this.
b) Not possible to skip the faulty loop. All non null arrays reach the faulty loop body.
c) x = [7, 2 ,9]; y = 2; Expected = -1; Actual = -1 → fault executed, but not classified as error state, because there are no matches in the array
d) x = [7, 8,9]; y = 2; Expected = -1; Actual = -1 → error state that did NOT result in failure, because expected and actual results match but, the program has no way of knowing if the first element was a 2. So this is an incorrect internal state only.
e) The first error state for test case d), is when the a 2 does not appear in any position other than the first index. This will not lead to a failure, but it is incorrect. It is wrong because the program can not inspect the first element to be sure that it was not a 2.

lastZero

a) This piece of code returns on the first instance of an integer 0 inside the array x. The function name and comments indicate it should return on the last instance of an integer 0. Keeping the return statement, with a comparison, outside the loop would fix this.
b) Not possible to skip the faulty comparison. All non null arrays reach the faulty comparison.
c) X = [1,1,0]; Expected = 1 ; Actual = 1 → fault executed, because the faulty comparison took place, but not classified as error state, since the faulty comparison is correct, in this case.
d) X = [1,0,1]; Expected = 1 ; Actual = 1 → error state that did NOT result in failure, because expected and actual match.
e) The first error state for test case on d), is when a 0 shows up in any position other than the last index. The return statement breaks out of the loop before completion. So the program has know way of knowing if any items after the first 0, were 0 themselves.

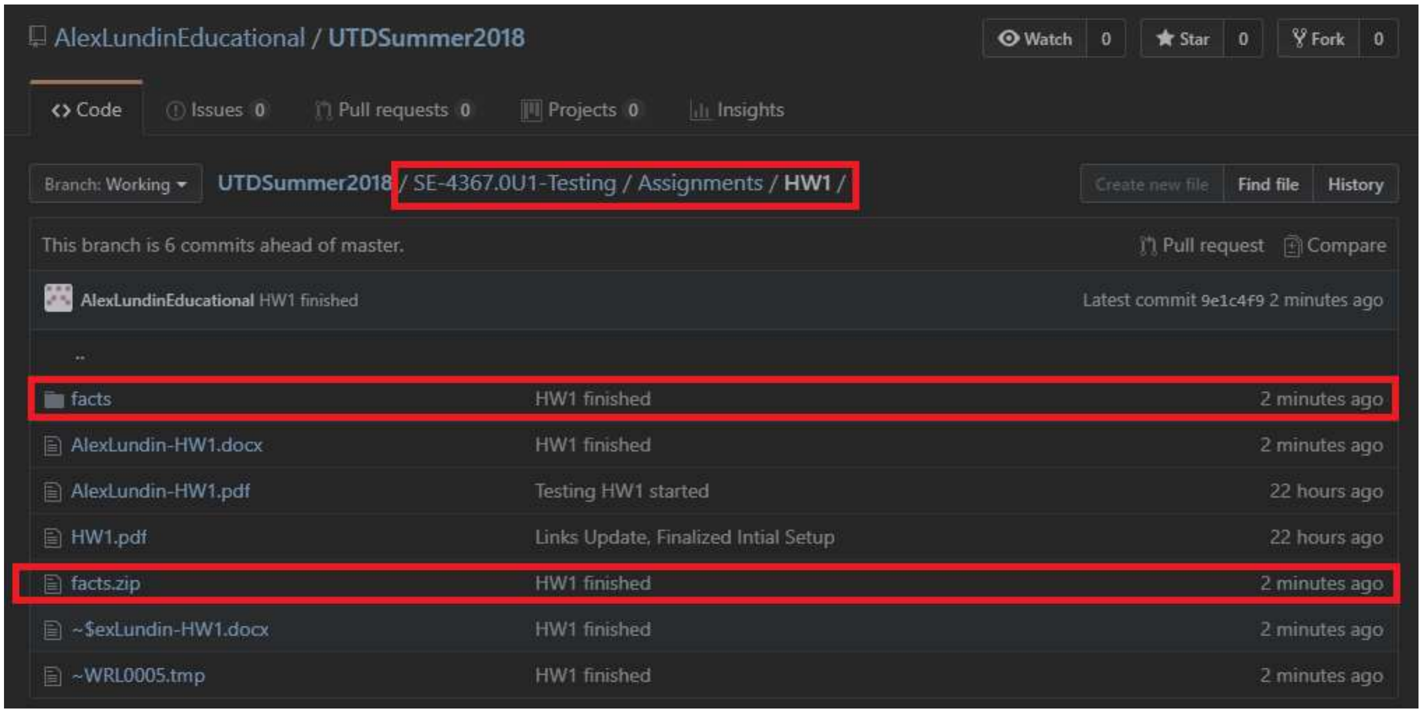
countPositive

a) This piece of code counts zeros as positive numbers. Changing the comparison to x[i] > 0 would fix this
b) Not possible to skip the faulty comparison. All non null arrays reach the faulty comparison.
c) X = [-4,2,1,2]; Expected = 3 ; Actual = 3 → fault executed, but not classified as error state, because there are no 0's to misclassify as positive numbers.
d) Not possible to execute the fault resulting in an error state without failure because 0's in the array cause error states, which always result in failures.
e) The first error state for test case d), is when any 0 is in the array, the counter increments, classifying 0 as a positive number.

oddOrPos

a) This piece of code doesn't account for negative odd numbers. Adding another or statement in the comparison would fix this.
b) Not possible to skip the faulty comparison. All non null arrays reach the faulty comparison.
c) X = [-2,-4,0,1,4]; Expected = 3 ; Actual = 3 → fault executed, but not classified as error state, because there are no negative odd numbers so expected and actual match.
d) Not possible to execute the fault resulting in an error state without failure because negative odd numbers execute the fault, which always results in a incorrect internal state finally leading to a failure.
e) The first error state for test case d), is when the input array has negative positive numbers in it.

3.) Facts zip storage on GitHub



AlexLundinEducational / UTDSummer2018

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Insights

Branch: Working UTDSummer2018 / SE-4367.0U1-Testing / Assignments / HW1 /

Create new file Find file History

This branch is 6 commits ahead of master. Pull request Compare

AlexLundinEducational HW1 finished Latest commit 9e1c4f9 2 minutes ago

..

facts	HW1 finished	2 minutes ago
AlexLundin-HW1.docx	HW1 finished	2 minutes ago
AlexLundin-HW1.pdf	Testing HW1 started	22 hours ago
HW1.pdf	Links Update, Finalized Intial Setup	22 hours ago
facts.zip	HW1 finished	2 minutes ago
~$exLundin-HW1.docx	HW1 finished	2 minutes ago
~WRL0005.tmp	HW1 finished	2 minutes ago

Homework1- (individual) **SE 4367: Software Testing, Verification, Validation and QA**

Student Name: Alex Lundin
Class and Section SE 4367.0U1
Total Points (Out of 100 points) ____________

Instruction:
1. Answer to the problem on a **PDF** file **(PDF file only)**
2. Submit the **PDF** file to eLearning before the due date

1. (5 points) Please **subscribe** to the Class Discussion Board for important announcements, questions, and updates.

2. (75 points) Below are four faulty programs. Each includes test inputs that result in failure. Answer the following questions about each program.

```
/**
 * Find last index of element
 *
 * @param x array to search
 * @param y value to look for
 * @return last index of y in x; -1 if absent
 * @throws NullPointerException if x is null
 */
public int findLast (int[] x, int y)
{
   for (int i=x.length-1; i > 0; i--)
   {
      if (x[i] == y)
      {
         return i;
      }
   }
   return -1;
}
// test:  x = [2, 3, 5]; y = 2; Expected = 0
// Book website: FindLast.java
// Book website: FindLastTest.java
```

```
/**
 * Find last index of zero
 *
 * @param x array to search
 *
 * @return last index of 0 in x; -1 if absent
 * @throws NullPointerException if x is null
 */
public static int lastZero (int[] x)
{
   for (int i = 0; i < x.length; i++)
   {
      if (x[i] == 0)
      {
         return i;
      }
   }
   return -1;
}
// test:  x = [0, 1, 0]; Expected = 2
// Book website: LastZero.java
// Book website: LastZeroTest.java
```

```
/**
 * Count positive elements
 *
 * @param x array to search
 * @return count of positive elements in x
 * @throws NullPointerException if x is null
 */
public int countPositive (int[] x)
{
   int count = 0;
   for (int i=0; i < x.length; i++)
   {
      if (x[i] >= 0)
      {
         count++;
      }
   }
   return count;
}
// test:  x = [-4, 2, 0, 2]; Expcted = 2
// Book website: CountPositive.java
// Book website: CountPositiveTest.java
```

```
/**
 * Count odd or postive elements
 *
 * @param x array to search
 * @return count of odd/positive values in x
 * @throws NullPointerException if x is null
 */
public static int oddOrPos(int[] x)
{
   int count = 0;
   for (int i = 0; i < x.length; i++)
   {
      if (x[i]%2 == 1 || x[i] > 0)
      {
         count++;
      }
   }
   return count;
}
// test:  x = [-3, -2, 0, 1, 4]; Expected = 3
// Book website: OddOrPos.java
// Book website: OddOrPosTest.java
```

(a) Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.
(b) If possible, give a test case that does not execute the fault. If not, briefly explain why not.

(c) If possible, give a test case that executes the fault, but does not result in an error state. If not, briefly explain why not.
(d) If possible give a test case that results in an error state, but not a failure. Hint: Don't forget about the program counter. If not, briefly explain why not.
(e) For the given test case, describe the first error state. Be sure to describe the complete state.

3. (20 points) Download the facts.zip file which contain the source files. It is a web application written in Java with a combination of Java servlets, Java server pages, JavaScript, and XML. We will be using it for future evolution and testing assignments. You will not need to be an expert in web apps to do these assignments, but you will need to understand how this small application works. For this assignment, all you need to do is download the files and examine them to see how well you can understand the code. You can download the files anywhere, but I recommend putting them in a repository like GitHub. **Submit a screen shot that shows where you put the facts files.**