Submission Guidelines:

Due: 11:59pm ending Wednesday, June 13, 2018.

- The assignment should be submitted via Blackboard.
- The answers must be typed as a document.
- Make sure your name and your student ID are listed in your document.
- Name files as assignment1_<net-id>.<format>
- Accepted document formats are (.pdf, .doc or .docx). If you are using OpenOffice or LibreOffice, make sure to save as .pdf or .doc
- Please do not submit .txt files.
- If there are multiple files in your submission, zip them together as assignment1_<net-id>.zip and submit the .zip file.
- The maximum points one can get in this assignment is 100.
- You may resubmit the project at any time. Late submissions will be accepted at a penalty of 10 points per day. Maximum latency is 5 days beyond which a grade of zero will be assigned. This penalty will apply regardless of whether you have other excuses.

CSE 5321 Software Testing - Assignment 1 – Summer 2018 Total points – 100

---

1. How are faults and failures related to testing and debugging? (5 points)

2. What is verification and validation? (5 points)

3. What is static analysis and dynamic testing? (5 points)

4. For what do testers use automation? What are the limitations of automation? (5 points)

5. Below are four faulty programs. Each includes a test case that results in failure. Answer the following questions about each program. (20 points each)

   (a) Identify the fault. (3 points)

   (b) If possible, identify a test case that does **not** execute the fault. (3 points)

   (c) If possible, identify a test case that executes the fault, but does **not** result in an error state. (3 points)

   (d) If possible identify a test case that results in an error, but **not** a failure. (3 points)
   Hint: Don't forget about the program counter.

   (e) For the given test case, identify the first error state. Be sure to describe the complete state. (5 points)

   (f) Fix the fault and verify that the given test now produces the expected output. (3 points)

Program 1:

```
public int findLast (int[] x, int y)
{
//Effects: If x==null throw NullPointerException
//   else return the index of the last element
//   in x that equals y.
//   If no such element exists, return -1
   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
```

Program 2:

```
public int countPositive (int[] x)
{
//Effects: If x==null throw NullPointerException
//   else return the number of
//        positive elements in 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]
   //        Expected = 2
```

Program 3:

```
public static int lastZero (int[] x) {
//Effects: if x==null throw NullPointerException
//   else return the index of the LAST 0 in x.
//   Return -1 if 0 does not occur in x

   for (int i = 0; i < x.length; i++)
   {
      if (x[i] == 0)
      {
         return i;
      }
   }
   return -1;
}
   // test:  x=[0, 1, 0]
   //        Expected = 2
```

Program 4:

```
public static int oddOrPos(int[] x) {
//Effects: if x==null throw NullPointerException
// else return the number of elements in x that
//      are either odd or positive (or both)
   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
```