Submission Guidelines:

Due: 11:59pm ending Wednesday, June 20, 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 assignment2_<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 submit at any time. Late submissions will be accepted at a penalty of 10 points per day. Maximum latency is 3 days beyond which a grade of zero will be assigned. This penalty will apply regardless of whether you have other excuses.

Assignment Specifications:

1. A tester defined four characteristics based on the input parameter car: **Where Made**, **Energy Source**, **Size,** and **Color**. The following partitioning for these characteristics have at least four mistakes. Correct them. (10 pts.)

Where Made	Energy Source	Size	Color
North America	Gas	2-door	White
Europe	Electric	4-door	Silver
Denmark	hybrid	Hatch back	Black
Africa			Blue

Hint: each partition must satisfy two properties (Completeness and disjointness).

2. Answer the following questions for the method search() below: (10 pts.)

```
public static int search (List list, Object element)
// Effects: if list or element is null throw NullPointerException
//   else if element is in the list, return an index
//   of element in the list; else return -1
//   for example, search ([3,3,1], 3) = either 0 or 1
//       search ([1,7,5], 2) = -1
```

Base your answer on the following characteristic partitioning:

```
Characteristic: Location of element in list
       Block 1: element is first entry in list
       Block 2: element is last entry in list
       Block 3: element is in some position other than first or last
```

a) "**Location of element in list**" fails the disjointness property. Give an example that illustrates this.
b) "**Location of element in list**" fails the completeness property. Give an example that illustrates this.
c) Supply one or more new partitions that capture the intent of "**Location of e in list**" but do not suffer from completeness or disjointness problems.

3. Derive input space partitioning tests for the **GenericStack** class with the following method signatures: (15 pts.)

- public GenericStack ();
- public void Push (Object X);
- public Object Pop ();
- public boolean IsEmt ();

Assume the usual semantics for the stack. Try to keep your partitioning simple, choose a small number of partitions and blocks

a) Define characteristics of inputs.
b) Partition the characteristics into blocks.
c) Define values for the blocks.

4. Answer the following questions for the method **intersection**() below: (15 pts.)

```
public Set intersection (Set s1, Set s2)
   // Effects:   If s1 or s2 are null throw NullPointerException
   //    else return a (non null) Set equal to the intersection
   //    of Sets s1 and s2

   Characteristic:  Type of s1
      - s1 = null
      - s1 = {}
      - s1 has at least one element

   Characteristic:  Relation between s1 and s2
      - s1 and s2 represent the same set
      - s1 is a subset of s2
      - s2 is a subset of s1
      - s1 and s2 do not have any elements in common
```

a) Does the partition "**Type of s1**" satisfy the completeness property? If not, give a value for s1 that does not fit in any block.
b) Does the partition "**Type of s1**" satisfy the disjointness property? If not, give a value for s1 that fits in more than one block.
c) Does the partition "**Relation between s1 and s2**" satisfy the completeness property? If not, give a pair of values for s1 and s2 that does not fit in any block.
d) Does the partition "**Relation between s1 and s2**" satisfy the disjointness property? If not, give a pair of values for s1 and s2 that fits in more than one block.
e) If the "**base choice**" criterion were applied to the two partitions (exactly as written), how many test requirements would result?

5. Derive input space partitioning tests for the **BoundedQueue** class with the following signature: (15 pts.)

- public BoundedQueue (int capacity);
- public void Enqueue (Object X);
- public Object Dequeue ();
- public boolean IsEmpty ();
- public boolean IsFull ()

Assume the usual semantics for a queue with a fixed, maximal capacity. Try to keep your partitioning simple–choose a small number of partitions and blocks.

a) Identify several characteristics that suggest partitions.
b) Identify the blocks in the partition for each characteristic. Designate one block in each partition as the "Base" block.
c) Define values for the blocks.
d) Define a test set that satisfies base choice coverage (BCC).

6. Write down all 64 tests to satisfy the All Combinations (ACoC) criterion for the second categorization of triang()'s inputs in Table 4.2. Use the values in Table 4.3. (15 pts.)

Table 4.2. Second partitioning of TriTyp's inputs (interface-based).

Partition	b_1	b_2	b_3	b_4
q_1 = "Length of Side 1"	*greater than 1*	*equal to 1*	*equal to 0*	*less than 0*
q_2 = "Length of Side 2"	*greater than 1*	*equal to 1*	*equal to 0*	*less than 0*
q_3 = "Length of Side 3"	*greater than 1*	*equal to 1*	*equal to 0*	*less than 0*

Table 4.3. Possible values for blocks in the second partitioning in Table 4.2

Param	b_1	b_2	b_3	b_4
Side 1	2	1	0	−1
Side 2	2	1	0	−1
Side 3	2	1	0	−1

7. Enumerate all 16 tests to satisfy the pair-wise (PWC) criterion for the second categorization of TriTyp's inputs in Table 4.2. Use the values in Table 4.3 (use the above tables). (10 pts.)

8. Enumerate all 16 tests to satisfy the multiple base choice (MBCC) criterion for the second categorization of TriTyp's inputs in Table 4.2. Use the values in Table 4.3 (use the above tables). (10 pts.)