Summer 2019 Midterm Solutions

```
1.      Method Call             Output Produced
        ------------------------------------------------
        mystery(9);             =9
        mystery(42);            24=
        mystery(703);           30=7
        mystery(5821);          12=58
        mystery(83105);         013=85
```

2. One possible solution appears below.

```
    public static boolean isPalindrome(String s) {
        if (s.length() <= 1) {
            return true;
        } else {
            return s.charAt(0) == s.charAt(s.length() - 1) &&
                    isPalindrome(s.substring(1, s.length() - 1));
        }
    }
```

```
3.  public Set<String> studyGroup(String topic, Map<String, Set<String>> skills) {
        Set<String> group = new TreeSet<String>();
        for (String person : skills.keySet()) {
            Set<String> bestSkills = skills.get(person);
            if (bestSkills.contains(topic)) {
                group.add(person);
            }
        }
        return group;
    }
```

```
4.
        before                    after                      code
-----------------------+-----------------------+-------------------------------
 p->[1]->[2]           | p->[1]->[3]           | ListNode temp = q;
                       |                       | q = p.next;
 q->[3]                | q->[2]                | p.next = temp;
-----------------------+-----------------------+-------------------------------
 p->[1]->[2]           | p->[1]->[3]           | q.next.next = p.next;
                       |                       | p.next = q;
 q->[3]->[4]           | q->[4]->[2]           | q = q.next;
                       |                       | p.next.next = null;
-----------------------+-----------------------+-------------------------------
                       |                       |
 p->[1]->[2]->[3]      | p->[3]->[2]           | p.next.next.next = p.next;
                       |                       | q = p;
                       |                       | p = p.next.next;
 q                     | q->[1]                | q.next = null;
                       |                       | p.next.next = null;
                       |                       |
                       |                       |
-----------------------+-----------------------+-------------------------------
```

```
-----------------------+-----------------------+-------------------------------
                       |                       |
                       |                       | p.next.next.next = q.next;
 p->[1]->[2]->[3]      | p->[1]->[2]->[3]      | ListNode temp = q;
                       |                       | q = p.next.next;
                       |                       | p.next.next = p;
                       |                       | p = p.next;
 q->[4]->[5]           | q->[3]->[5]           | p.next.next = temp;
                       |                       | temp.next = null
                       |                       |
-----------------------+-----------------------+-------------------------------
```

5. One possible solution appears below.

```
    public boolean isConsecutive(Stack<Integer> s) {
        if (s.size() <= 1)
            return true;
        else {
            Queue<Integer> q = new LinkedList<Integer>();
            int prev = s.pop();
            q.add(prev);
            boolean ok = true;
            while (!s.isEmpty()) {
                int next = s.pop();
                if (prev - next != 1)
                    ok = false;
                q.add(next);
                prev = next;
            }
            while (!q.isEmpty())
                s.push(q.remove());
            while (!s.isEmpty())
                q.add(s.pop());
            while (!q.isEmpty())
                s.push(q.remove());
            return ok;
        }
    }
```

6. One possible solution appears below.

```
    public void retainAll(Set<Integer> s) {
        for (int i = 0; i < size; i++) {
            if (!s.contains(elementData[i])) {
                for (int j = i; j < size - 1; j++) {
                    elementData[j] = elementData[j + 1];
                }
                size--;
                i--;
            }
        }
    }
```