

Solution to CSE143X Section #18 Problems

1.	Statement	Output
	-----	
	var1.method2 ();	Second2
	var2.method2 ();	Third2
	var3.method2 ();	Fourth2
	var4.method2 ();	Third2
	var5.method2 ();	compiler error
	var6.method2 ();	compiler error
	var1.method3 ();	Second2
	var2.method3 ();	Third2
	var3.method3 ();	Fourth2
	var4.method3 ();	Third2
	var5.method3 ();	compiler error
	var6.method3 ();	compiler error
	((Second)var4).method1 ();	compiler error
	((Third)var4).method1 ();	Third1/Second2
	((Second)var5).method2 ();	runtime error
	((First)var5).method3 ();	Fourth2
	((Third)var5).method1 ();	runtime error
	((First)var6).method3 ();	Second2
	((Second)var6).method1 ();	compiler error
	((Second)var6).method3 ();	Second2
	-----	
2.	Statement	Output
	-----	
	var1.method1 ();	harry 1
	var2.method1 ();	larry 1/harry 1
	var3.method1 ();	larry 1/harry 1
	var4.method1 ();	larry 1/harry 1
	var5.method1 ();	larry 1/harry 1
	var6.method1 ();	compiler error
	var1.method2 ();	harry 1/harry 2
	var2.method2 ();	larry 1/harry 1/harry 2
	var3.method2 ();	mary 2/jerry 2
	var4.method2 ();	mary 2
	var5.method2 ();	mary 2/jerry 2
	var6.method2 ();	compiler error
	var3.method3 ();	compiler error
	var5.method3 ();	larry 1/harry 1/mary 3
	((Larry)var1).method1 ();	runtime error
	((Mary)var2).method2 ();	runtime error
	((Jerry)var5).method1 ();	larry 1/harry 1
	((Mary)var3).method3 ();	larry 1/harry 1/mary 3
	((Jerry)var4).method3 ();	runtime error
	((Mary)var6).method3 ();	runtime error

3.	Statement	Output
	var1.method1 ();	foo 1/bar 2
	var2.method1 ();	baz 1/mumble 1
	var3.method1 ();	foo 1/bar 2
	var4.method1 ();	baz 1
	var5.method1 ();	baz 1/mumble 1
	var6.method1 ();	compiler error
	var1.method2 ();	bar 2
	var2.method2 ();	baz 2/baz 1/mumble 1
	var3.method2 ();	bar 2
	var4.method2 ();	baz 2/baz 1
	var5.method2 ();	baz 2/baz 1/mumble 1
	var6.method2 ();	compiler error
	var3.method3 ();	bar 3
	var5.method3 ();	compiler error
	((Bar)var1).method3 ();	bar 3
	((Mumble)var4).method3 ();	runtime error
	((Mumble)var5).method3 ();	mumble 3
	((Bar)var2).method3 ();	runtime error
	((Baz)var2).method2 ();	baz 2/baz 1/mumble 1
	((Mumble)var6).method2 ();	runtime error

4.	Statement	Output
	var1.method2 ();	Fo 2/Fie 3/Fum 3
	var2.method2 ();	compiler error
	var3.method2 ();	Fo 2/Fee 3
	var4.method2 ();	compiler error
	var5.method2 ();	compiler error
	var6.method2 ();	Fo 2/Fum 3
	var1.method3 ();	Fie 3/Fum 3
	var2.method3 ();	compiler error
	var3.method3 ();	Fee 3
	var4.method3 ();	compiler error
	var5.method3 ();	compiler error
	var6.method3 ();	Fum 3
	((Fee)var3).method1 ();	Fee 1/Fo 3
	((Fee)var4).method1 ();	runtime error
	((Fie)var1).method2 ();	Fo 2/Fie 3/Fum 3
	((Fo)var3).method3 ();	Fee 3
	((Fie)var6).method2 ();	runtime error
	((Fo)var2).method3 ();	Fum 3
	((Fie)var3).method1 ();	runtime error
	((Fo)var5).method2 ();	Fo 2/Fo 3

5. One possible solution appears below.

```
public void splitStack(Stack<Integer> s) {
    Queue<Integer> q = new LinkedList<>();
    int oldLength = s.size();
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
    for (int i = 1; i <= oldLength; i++) {
        int n = q.remove();
        if (n < 0) {
            s.push(n);
        } else {
            q.add(n);
        }
    }
    while (!q.isEmpty()) {
        s.push(q.remove());
    }
}
```

6. One possible solution appears below.

```
public Stack<Integer> copyStack(Stack<Integer> s) {
    Stack<Integer> s2 = new Stack<>();
    Queue<Integer> q = new LinkedList<>();
    while (!s.isEmpty()) {
        s2.push(s.pop());
    }
    while (!s2.isEmpty()) {
        q.add(s2.pop());
    }
    while (!q.isEmpty()) {
        int n = q.remove();
        s.push(n);
        s2.push(n);
    }
    return s2;
}
```

7. One possible solution appears below.

```
public boolean equals(Stack<Integer> s1, Stack<Integer> s2) {
    if (s1.size() != s2.size()) {
        return false;
    } else {
        Stack<Integer> s3 = new Stack<>();
        boolean same = true;
        while (same && !s1.isEmpty()) {
            int num1 = s1.pop();
            int num2 = s2.pop();
            if (num1 != num2) {
                same = false;
            }
            s3.push(num1);
            s3.push(num2);
        }
        while (!s3.isEmpty()) {
            s2.push(s3.pop());
            s1.push(s3.pop());
        }
        return same;
    }
}
```

8. One possible solution appears below.

```
public static void collapse(Stack<Integer> s) {
    Queue<Integer> q = new LinkedList<>();
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
    while (!q.isEmpty()) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
    while (q.size() > 1) {
        s.push(q.remove() + q.remove());
    }
    if (!q.isEmpty()) {
        s.push(q.remove());
    }
}
```

9. One possible solution appears below.

```
public void rearrange(Queue<Integer> q) {
    Stack<Integer> s = new Stack<>();
    int oldSize = q.size();
    for (int i = 0; i < oldSize; i++) {
        int n = q.remove();
        if (n % 2 == 0) {
            q.add(n);
        } else {
            s.push(n);
        }
    }
    int evenCount = q.size();
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
    for (int i = 0; i < evenCount; i++) {
        q.add(q.remove());
    }
    for (int i = 0; i < oldSize - evenCount; i++) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
}
```

10. One possible solution appears below.

```
public boolean isPalindrome(Queue<Integer> q) {
    Stack<Integer> s = new Stack<>();
    for (int i = 0; i < q.size(); i++) {
        int n = q.remove();
        q.add(n);
        s.push(n);
    }
    boolean ok = true;
    for (int i = 0; i < q.size(); i++) {
        int n1 = q.remove();
        int n2 = s.pop();
        if (n1 != n2) {
            ok = false;
        }
        q.add(n1);
    }
    return ok;
}
```

11. One possible solution appears below.

```
public void shift(Stack<Integer> s, int n) {
    if (n < 0 || n > s.size()) {
        throw new IllegalArgumentException();
    }
    Queue<Integer> q = new LinkedList<>();
    int otherSize = s.size() - n;
    for (int i = 0; i < otherSize; i++) {
        q.add(s.pop());
    }
    while (!q.isEmpty()) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
    while (!q.isEmpty()) {
        s.push(q.remove());
    }
}
```

12. One possible solution appears below.

```
public void reverseByN(Queue<Integer> q, int n) {
    Stack<Integer> s = new Stack<>();
    int times = q.size() / n;
    int extra = q.size() % n;
    for (int i = 0; i < times; i++) {
        for (int j = 0; j < n; j++) {
            s.push(q.remove());
        }
        while (!s.isEmpty()) {
            q.add(s.pop());
        }
    }
    for (int i = 0; i < extra; i++) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
}
```