A. Black and White Box Tests

1. Getting started:

Leider haben wir den Fehler gemacht und uns den Code vorher angeschaut. Wir haben natürlich sofort gemerkt das es einen falsches Ergebnis für den Wert "-1" gibt und haben unsere Tests dann auch sofort so angelegt dass sie die "gewünschten" Fehlermeldungen ausspucken.

```
public class TestAbsolute {
    public static int absoluteValueOf(int x) {
        if (x < -1)
            return -x;
        else
            return x;
    }

    @Test
    public void testFail() {
        int x = absoluteValueOf(-1);
        assertTrue(x == 1);
    }

    @Test
    public void testSuccess() {
        int x = absoluteValueOf(-2);
        assertTrue(x == 2);
    }
}
```

2. Black-box test:

Da Sie uns schon in der Übung den Lösungsweg gezeigt haben war diese Aufgabe nicht wirklich ein Problem.

Hier der Input:

```
96
91
86
81
76
71
66
61
56
51
46
41
36
31
26
21
16
11
6
1
-1
```

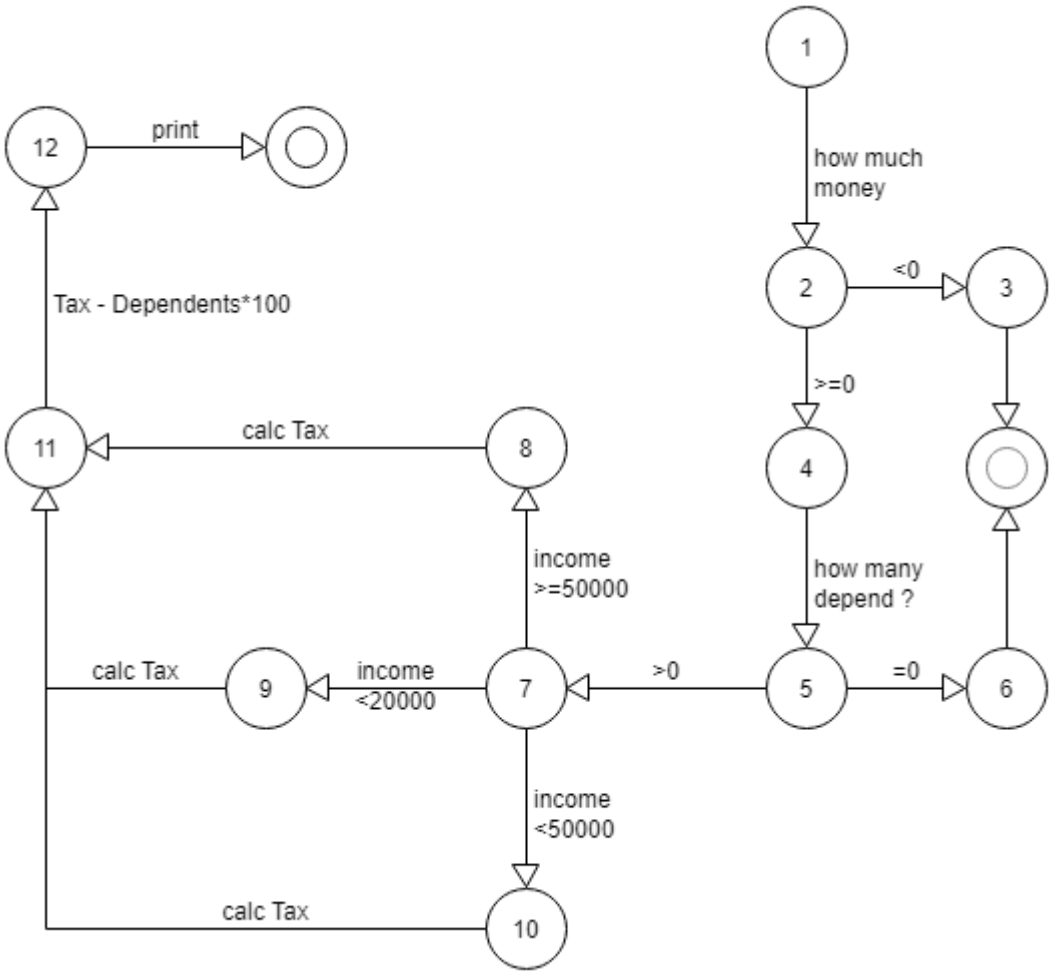
Hier der Output:

```
Letter grade: A
Letter grade: A
Letter grade: B
Letter grade: B
Letter grade: B
Letter grade: C
Letter grade: C
Letter grade: C
Letter grade: D
Letter grade: D
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Letter grade: F
Average: 46.19047619047619
```

3. White-box test:

Wir wussten leider nicht wie man einen Junit Test für Konsolen-Applikationen entwirft weswegen wir es “per hand” (über die Konsole) getestet haben.

Graph:



Unserer Meinung gibt es 5 Wege um zum “Ende” des Programmes zu kommen. Folgend dafür sind die Gründe dafür tabellarisch aufgeführt:

Conditions	**Current state**	**Next state**	**Consequence**
Income < 0	2	3	Exit
Dependents <= 0	3	6	Exit
Income >= 50000	7	8	Calc Tax
Income < 50000	7	10	Calc Tax
Income < 20000	7	9	Calc Tax

Test 1 (Income < 0):

Welcome to the new Berlin tax calculator.
How much did you earn last year? -1
Even in Berlin, no one has a negative income!
Start over.

Test 2 (Dependents <= 0):

Welcome to the new Berlin tax calculator.
How much did you earn last year? 1
Enter the number of dependents you have, including yourself: 0
Did you forget to count

Test 3 (Income >= 50000):

Welcome to the new Berlin tax calculator.
How much did you earn last year? 10000000
Enter the number of dependents you have, including yourself: 2
=€=€=€=€=€=€=€=€=€=€=€=€=€=€=€
Wowereit & Sarrazin GmbH
Tax bill
Your income was 1.0E7 €.
You have 2 family members.
Your total tax is 3583200.0 €.
Family member tax saving is 200€.
=€=€=€=€=€=€=€=€=€=€=€=€=€=€=€

Test 4 (Income < 50000):

Welcome to the new Berlin tax calculator.
How much did you earn last year? 49000
Enter the number of dependents you have, including yourself: 2
=€=€=€=€=€=€=€=€=€=€=€=€=€=€=€
Wowereit & Sarrazin GmbH
Tax bill
Your income was 49000.0 €.
You have 2 family members.
Your total tax is 9360.0 €.
Family member tax saving is 200€.
=€=€=€=€=€=€=€=€=€=€=€=€=€=€=€

Test 5 (Income < 20000):

```
Welcome to the new Berlin tax calculator.
How much did you earn last year? 19000
Enter the number of dependents you have, including yourself: 2
=€=€=€=€=€=€=€=€=€=€=€=€=€=€=€
Wowereit & Sarrazin GmbH
Tax bill
Your income was 19000.0 €.
You have 2 family members.
Your total tax is 2160.0 €.
Family member tax saving is 200€.
=€=€=€=€=€=€=€=€=€=€=€=€=€=€=€
```

Test 6 (Input = NaN):
1. (Income = NaN):

```
Welcome to the new Berlin tax calculator.
How much did you earn last year? f
Enter the number of dependents you have, including yourself:
java.lang.NumberFormatException: For input string: "f"
        at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2043)
        at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.lang.Double.parseDouble(Double.java:538)
        at TaxTime.main(TaxTime.java:28)
```

2. (Dependents = NaN):

```
Welcome to the new Berlin tax calculator.
How much did you earn last year? 1
Enter the number of dependents you have, including yourself: d
Exception in thread "main" java.lang.NumberFormatException: For input string: "d"
        at
java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.valueOf(Integer.java:766)
        at TaxTime.main(TaxTime.java:46)
```

Fehler die wir “gefunden” haben:

Als Studenten haben wir leider keinerlei Ahnung von Steuern :-).
Dennoch sollte ein Steuerberechnungs-Programm Inputs die keine Zahlen sind verwerten können und “anständige” Fehlermeldungen geben anstack einfach einen Stacktrace auszuwerfen.

4. Reflection:

Wie schon oben angemerkt ist es eher schwierig eine Konsolen-Applikation über automatische Tests zu testen bzw. zu entwickeln. Unserer Meinung nach wäre eine Applikation mit return Werten und gegebenen Falls einer grafischen Benutzer-Oberfläche nötig um diese Applikation “sinnvoll” zu gestalten.

B. Test Driven Development

An sich war dies eigentlich eine sehr einfache Aufgabe. Das größte Hindernis war allerdings die Aufgabenstellung zu entziffern, denn in der Aufgabenstellung stand dass wir die Klasse Node entickeln sollen, welche aber bereits vorgegeben war. Wir haben also eine Klasse LinkedList entworfen (Da dies mehr Sinn gemacht hat).

Hier die Klasse:

```
public class LinkedList<E> {
	public Node<E> head;

	public LinkedList() {
		head = null;
	}
	public void add(Node<E> node) {
		Node<E> current = head;
		if (head == null) {
			head = node;
			return;
		}
		while (current.next != null) {
			current = current.next;
		}
		current.next = node;
	}
	public void reverse() {
		Node<E> next;
		Node<E> current = head;
		Node<E> prev = null;
		while (current != null) {
			next = current.next;
			current.next = prev;
			prev = current;
			current = next;
		}
		head = prev;
	}
	public void delete(E key) {
		Node<E> temp = head;
		Node<E> prev = null;
		if (temp != null && temp.data == key) {
			head = temp.next;
			return;
		}
		if (temp == null) {
			return;
		}
		while (temp != null && temp.data != key) {
			prev = temp;
			temp = temp.next;
		}
		prev.next = temp.next;

	}

	public String toString() {
		String result = "";
		Node<E> current = head;
		do {
		current = current.next;
		result += current.data + ",";
		}
		while (current.next != null);

		return result;
		//alles ausser Head wird ausgegeben
	}
}
```

Hier die Tests:

```
class test1 {
	@Test
	void add() {
		LinkedList<String> l = new LinkedList<>();
		l.add(new Node<>("A"));
		System.out.println(l.head.data);
		assertTrue(l.head.data == "A" );
	}
	@Test
	void add2() {
		LinkedList<String> l = new LinkedList<>();
		l.add(new Node<>("A"));
		l.add(new Node<>("B"));
		System.out.println(l.head.data);
		assertTrue(l.head.next.data == "B" );
	}
	@Test
	void removeA() {
		LinkedList<String> l = new LinkedList<>();
		l.add(new Node<>("A"));
		System.out.println("_____");
		System.out.println(l.head.data);
		l.delete("A");
		System.out.println(l.head);
		assertTrue(l.head == null);
	}
	@Test
	void removeB() {
		LinkedList<String> l = new LinkedList<>();
		l.add(new Node<>("A"));
		l.add(new Node<>("B"));
		l.add(new Node<>("C"));
		System.out.println("_____");
		System.out.println(l.head.data);
		l.delete("B");
		System.out.println(l.head);
		assertTrue(l.head.next.data == "C");
	}
	@Test
	void reverse() {
		LinkedList<String> l = new LinkedList<>();
		l.add(new Node<>("A"));
		l.add(new Node<>("B"));
		l.add(new Node<>("C"));
		l.add(new Node<>("D"));
		l.add(new Node<>("E"));
		System.out.println("_____");
		System.out.println(l.head.data);
		l.reverse();

		System.out.println(l.head.data);
		System.out.println(l.head.next.data);
		System.out.println(l.toString());
		assertTrue(l.head.data == "E");
	}
```