Testing und Debugging - Übungsblatt 8:

Gruppe 11:
Manuel Neubauer - 79358
Hueseyin Selcuk - 79375
Abdullah Yildiz - 79669
Daniel Zahl - 79356

1. **Systematische Fehlersuche**

    1. Vervollständigen Sie die folgende Tabelle:

Programmschritt		Variablen				
Schritt	Zeile	s	tag	quote	out	c
1	def remove_html_markup(“5>3”)					
		5>3	?	?	?	?
2	tag = False					
		5>3	False	?	?	?
3	quote = False					
		5>3	False	False	?	?
4	out = “”					
		5>3	False	False	“”	?
5 Schleife 1	for c in s:					
		5>3	False	False	“”	5
6	if c == '<' and not quote:					
		5>3	False	False	“”	5
7	elif c == '>' and not quote:					
		5>3	False	False	“”	5
8	elif (c == "' or c == """) and tag:					
		5>3	False	False	“”	5
9	elif not tag:					
		5>3	False	False	“”	5
10	out = out + c					
		5>3	False	False	5	5
11 Schleife 2	for c in s:					
		5>3	False	False	5	>
12	if c == '<' and not quote:					
		5>3	False	False	5	>
13 **Defekt**	elif c == '>' and not quote: **Defekt**					
		5>3	False	False	5	>
14	elif (c == "' or c == """) and tag:					
		5>3	False	False	5	>
15	elif not tag: D					
		5>3	False	False	5	>
16	out = out + c					
		5>3	False	False	5	>
17 Schleife 3	for c in s: **Infektion**					
		5>3	False	False	5 **Verbreitung**	3
18 **Verbreitung**	if c == '<' and not quote:					
		5>3	False	False	5 **Verbreitung**	3
19 **Verbreitung**	elif c == '>' and not quote:					
		5>3	False	False	5 **Verbreitung**	3

20 **Verbreitung**	elif (c == "' or c == """) and tag:					
		5>3	False	False	5 **Verbreitung**	3
21 **Verbreitung**	elif not tag:					
		5>3	False	False	5 **Verbreitung**	3
22 **Verbreitung**	out = out + c					
		5>3	False	False	53 **Fehlverhalten**	3
23	return out					

2. In welcher der folgenden Aussagen treffen zu (4 + 3 + 3 Punkte):

1. Jede Infektion endet in einem Fehlverhalten.
FALSE
2. Jede Infektion kann zu einem Defekt zurückverfolgt werden, der die Infektion verursacht hat.
TRUE
3. Jeder Defekt endet in einem Fehlverhalten.
FALSE

Zu Aussage 1:

Infektionen müssen nicht zwangsweise in einem Fehlverhalten enden.
Wenn die Ausgabe nicht in direktem Zusammenhang mit der Infektion steht,
kann die Ausgabe bzw. das Verhalten der Software trotzdem korrekt sein.

Zu Aussage 3:

Ein Defekt (eine Schadhafte Stelle im Programm) führt zu Fehlern in einem Programmzustand.
Wenn dieser Programmzustand aber nicht verwendet wird, führt es zu keinem Fehlverhalten.

3. Markieren Sie in der Tabelle aus 1.1.1 den Defekt, die Infektion und zeigen Sie, wie sich die Infektion bis zum Fehlverhalten hin ausbreitet. (10 Punkte):

   Siehe Tabelle 1.1

4. Wie ist der Defekt zu beheben?:

   Um den Defekt zu beheben muss in Zeile 8 geprüft werden, ob bereits tag true ist.
   Ist Tag false wird “>” als normales character gewertet und dem out string angefügt:

   elif c == ‘>’ and not quote and tag

2. Fixen Sie das Fehlverhalten für die Eingabe <input value="5’>’3"> (50 Punkte):

1. Finden Sie mit dieser Methode die Ursache dieses Fehlverhaltens. (40 Punkte):

Hypothese:
durch das > Zeichen innerhalb des Value Attributes wird dies als tag-ende Symbol interpretiert.
Dadurch wird der Rest als String ausgegeben obwohl dieser eigentlich noch zu dem html tag dazugehört.

Experiment:

Eingabe:	**Erwartet:**	**Ausgabe:**
<input value="5’’3">	5>3	''
<>input value="5’>’3">		'input value="5’>’3">'

Durch das entfernen des > Zeichens in input 1 wird dies nur als ein tag interpretiert. somit ist die Ausgabe ein leerer String.
Durch das hinzufügen des > Zeichens in input2 wird der Rest als String ausgegeben, so wie in der Hypothese vermutet.

Hypothese2:
Ein quote welches mit " beginnt, kann Fälschlicherweiße mit einem ' beendet werden. Und umgekehrt.

Experiment:

Eingabe:	**Erwartet:**	**Ausgabe:**
<img url="beeder>'2.mp3">		''
<img url="beeder'>2.mp3">		'2.mp3">'

Hypothese3:
Wir merken uns, mit welchem quote symbol (" oder ') die quote begonnen wurde.
Folgt ein ' auf ein mit " geöffneten qoute wird diese somit nicht Fälschlicherweiße beendet.

Experiment:

Eingabe:	**Erwartet:**	**Ausgabe:**
<tag> “5'>’3” </tag>	5>3	5>3

2. Durch welches Code Coverage Verfahren, dass Sie im ersten Teil der Vorlesung kennen gelernt haben, können Sie sicher Stellen, dass die Bugfixe für diese beiden Probleme vollständig getestet worden sind? Begründen Sie Ihre Antwort (10 Punkte):

   Da die Bugs sich in den if-elif Statements befinden, bietet sich hier der MC/DC Coverage an. Dieser prüft jede Bedienung unabhängig von den anderen Teilbedienungen.