Homework 3

To: All students in CSCI 5802 (Spring 2018)
CC: Teaching Assistant
From: Instructor
Date: 2/21/2018
Due: 3/5/2018, Monday, 11:55 PM, on Moodle.
Re: Homework Assignment 3 – Finite State Verification, Structural Testing, Data Flow Testing

---

There are 3 problems worth a total of 100 points. You may discuss these problems in your teams and turn in a single submission for the team on Moodle. You are supposed to turn in three files:

- Written answers for all questions in PDF format.
- A NuSMV source file, which contains your finite state model and all properties you wrote for Question 1.
- A NuSMV output file, which contains the results of verifying your properties on your system using NuSMV.

PROBLEM 1. 50 Points

For this exercise, you are required to create a finite-state model of a traffic-lights controller and verify its properties using the NuSMV symbolic model-checker.

- Assume that the controller manages traffic and pedestrian lights at the intersection of two roads, both with two-way traffic.
- Pedestrians can request access to cross the road by pressing a "walk button".
- Assume that the system has traffic sensors for each direction to detect if vehicles are present and waiting to pass through, which allows the system to manage traffic flow efficiently by varying the amount of time the lights are green for each road/direction based on demand. Your model should capture and represent this notion of *varying time* in some manner (i.e., **do not abstract away time**).
- There are emergency vehicle sensors for each direction, which lets the system provide priority access for emergency vehicles by switching lights appropriately.

You may state and make any other reasonable simplifying assumptions that you need. In your submission, you must address the following:
1. (10 Points) Define the scope and the requirements for the system that you intend to model – a brief description of what you have modelled, any assumptions that you have made and the key requirements you expect the system to satisfy.

2. (14 Points) Build a finite state model of the system. A NuSMV input source with sufficient comments would be adequate. Submit this source file on Moodle. (Though not required, you may find drawing state diagrams helpful).

3. (8 Points) Write at least two safety properties in temporal logic (CTL or LTL) that must be satisfied by the system. Explain your properties informally and state which system requirements those properties are derived from.

4. (8 Points) Write at least two liveness properties in temporal logic (CTL or LTL) that must be satisfied by the system. Explain your properties informally and state which system requirements those properties are derived from.

5. (5 Points) Write at least one "anti-property" – i.e., a temporal property that does not hold for the system – and show the counter-example generated by NuSMV for the anti-property. Briefly explain both your anti-property and the counter-example informally.

6. (5 Points) Verify all your properties on your system using the NuSMV symbolic model checker and provide a transcript of your NuSMV session. Submit this transcript file on Moodle.

Problem 2 (24 points)

For the C program shown at the end of this document:

1. Provide a test suite that achieves statement coverage, but not branch coverage.

2. Provide a test suite that achieves branch coverage, but not condition coverage. (Treat compound conditions as a single branch point for this purpose.)

3. Provide a test suite that achieves condition and branch coverage. What additional test cases, if any, do you need for Modified Condition/Decision Coverage (MC/DC)?

If any of these test suites are not feasible, justify why it is the case. Feel free to try executing the code (no guarantees here), with any additions that you may need, to check whether your tests achieved the desired level of structural coverage.

Problem 3 (26 points)

a. (18 points) Exercise 13.1 in textbook. For part (b) compare the test suites for the new criteria with that for the All DU-pairs criterion and explain whether you would need fewer or more test cases for the all the variables in the routine. Provide justifications for all your claims.

b. (8 points) Exercise 13.2 in textbook. Show the hierarchy of subsumes relations among the five criteria (including the two defined in the previous problem) and briefly argue why each of those relations is correct.

```c
#include <stdio.h>

#define SWAP(a,b) { int _tmp = a; a = b; b = _tmp; }

int cur_count = 0, alloc_count = 0, total_count = 0, num_elems = 0;
int *counts, *psums, *elems;

void initialize(int n, FILE *fptr) {
	int i = 0;
	num_elems = n;
	counts = (int *) calloc(n, sizeof(int));
	psums = (int *) calloc(n, sizeof(int));
	elems = (int *) calloc(n, sizeof(int));

	for (i = 0; i < n; ++i) {
		elems[i] = i + 1;
		fscanf(fptr, "%d", &counts[i]);
		psums[i] = total_count += counts[i];
	}
}

int next_elem() {
	int top = num_elems - 1;
	int ret = elems[top];

	if (alloc_count >= total_count)
		return 0;

	++alloc_count;
	++cur_count;
	--psums[top];
	--counts[top];

	while (top > 0 && counts[top] <= (cur_count * psums[top - 1])) {
		psums[top - 1] += (counts[top] - counts[top - 1]);
		SWAP(counts[top], counts[top - 1]);
		SWAP(elems[top], elems[top - 1]);
		--top;
	}

	if (top != num_elems - 1)
		cur_count = 0;

	return ret;
}

int main() {
	int next;
	printf("number of elements: ");
	scanf("%d", &num_elems);
	printf("\nweights:\n");

	initialize(num_elems, stdin);

	printf("\n");
	while ((next = next_elem()) > 0)
		printf("%d ", next);
	printf("\n");
}
```