# CS132 HW3 Grading

## 1. Model

### Customer template – 3 points.

### Server template – 1 point.

The maximum response time is 1min, which means for the least patient customer, the server has to show up as quickly as possible. You can also say this time does not exist, if you think the least patient customer can get angry exactly at 1 minute. (Both answers are right)

### Chef template – 1 point.

The chef's maximum response time is 10 min.

## 2. Verifier query (The answer may vary with specific implementations)

(1) The customer should never leave without eating: **(2 points)**

(A[]) Customer.leave imply (Customer.eat == true)

(2) The customer always eventually eats: **(1 point)**

A<> Customer.Eat

(3) The customer should never be angry: **(1 point)**

A[] Customer.angry == false

If your answer to server minimum response time is "not exist", you may not pass property (3). Otherwise, you have to pass all three of them.

## 3. Organization of automata graph (1 point)

You will get a 0.5 point deduction in this part if your automata graph looks messy, or being confusing in some ways.

Common problems for this HW:

**(1) Send two "Order!" signal in server template.**

Please use broadcast channel when you need multiple receivers. There's no need to distinguish the signals sent to chef and customer.

**(2) Strange operations to the clocks.**

Generally you shouldn't perform anything to the clock except reset or synchronization. Use bool variables wisely.

**(3) Extra queries.**

Technically you can easily represent the properties mentioned in Q2 in one query. Extra queries will cause trouble for traceability. You will get point deduction for use more than one queries to verify one property unless the relation between them is clear. (Remember you will be explaining your model to non-professionals in your traceability report, make your verification nice and clean.)

**(4) Stuck at initial state**

You should force the customer leave his/her initial state by add invariables, otherwise you

may not satisfy A<> Customer.Eat property.

**(5) Failed to verify Q2.(1).**

It's wrong to verify this property by A<> Customer.Leave, unless "Leave" represents the specific state that the customer becomes angry and then leave. If your customer model always end in "Leave" state, it's unnecessary to verify A<> Customer.Leave since it naturally holds.