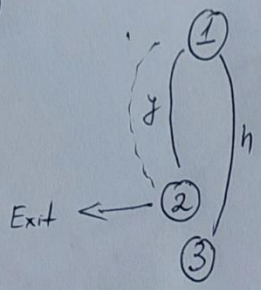
Input to the ticketing function is a booking object with the following properties:

- User id – used to find a user object. In the test cases I will use a user object for clarity.
- withChild – a Boolean variable that states whether the user is travelling with a child.
- departureTime – the time of the train departure, which is used to determine non-rush hour discount.
- 1….n destinationId parameters – these are used to create a list of destinations, where the first one is the departure destination and the last one is the arrival destination. Providing one means that the trip is void and will result in a price of zero, whereas providing more than two destinations results in a multi way trip.

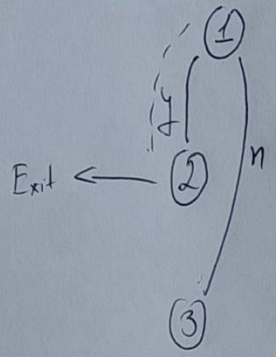
1. Inputs for statement coverage
    - {…userInfo, type: FAMILY}, true, 10.30, 1,2
    - {…userInfo, type: FAMILY}, true, 8.30, -1 (result in a faulty ID)
    - {…userInfo, type: NONE}, true, 10.30, 1,2
    - {…userInfo, type: OVER_60}, false, 10.30, 1,2
    - {}, false, 8.30, 1,2
2. Decision coverage
    - {}, false, 8.30, 1,2
    - {…userInfo, type: NONE}, true, 8.30, -1
    - {…userInfo, type: FAMILY}, true, 10.00, 1,2,3
    - {…userInfo, type: OVER_60}, false, 8.30, 1,2
    - {…userInfo, type: NONE}, true, 11.00, 1,2
    - {…userInfo, type: null}, …
3. Condition coverage
    - {…userInfo, type: null}, …
    - {}, …
    - …, -1
    - …, [] -> no destinations provided.
    - {…userInfo, type: OVER_60}, false, 10.30, 1,2
    - {…userInfo, type: FAMILY}, true, 8.30, 1,2
    - {…userInfo, type: FAMILY}, false, 10.30, 1,2
    - {…userInfo, type: NONE}, true, 17.30, 1,2

4. Graphs

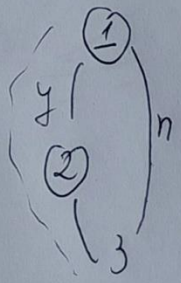
1) if(user==null)



2) if (destination List. length == 0 // invalid id)



3) if (!rushHours)



..... statement

decision ——

4) discount logic