Jaspal Khanuja

UCLA ID: 105864172

Mr. Howard

Project #3

In this project, I believe that I took a more conceptual and complicated design of the elevator. To further evaluate, I added a multitude of variables in my code such as moving, last\_value, current\_passengers, and more. For my project I believe the isValidElevatorString was the hardest function to create by far. This is because the way I created this was through a whole system of if and else statements within for loop which ran for very specific data depending on the character and the character before it. It used variables like moving to signify to the function that if ‘O’ has been stated, then and only then can C, +, - be stated. This in turn took a very long time to make perfect as there were definitely defects that legitimately took hours and hours to find. But I have to admit, if it weren't for this project, I don't think I would have ever discovered how incredibly useful the debugger is on xcode. It goes line by line of how commands are executed and I honestly do not believe I could have done this project without it. Some highly important features I learnt were: implementing last value, using static variables so that I could change them only when a number got greater than it, using string functions like find to help with my doorOpen function. Here are some parts of this code for reference. To start, for the elevator string I added this is some pseudo code: “if current passengers is == 0 and the last value is equal to -, then return false and break the loop. If the number of current passengers at any point is less than 0, then return false. If the string size of the elevator string is less than2 , then return false. If i ==0, then change last\_value = the current value which would be i==0, the current value would then be i== 1. If the last value is equal to M, and the current value is a digit. Then update the last value to a digit and continue the loop. For my other functions I incorporated similar parts of code that ran similar things such as the most passengers. It used the same type of code except at the end of every update for the current passengers if most passengers is < than the current passengers, then update most passengers to the new current value. Here is the actual parts of code that I described:

**for**(**int** i = 0; i < elevatorString.size(); i++){

**if**(current\_passengers == 0 && last\_value == '-'){

**return** **false**;

**break**;

}

**if** (current\_passengers < 0) {

**return** **false**;

**break**;

}

**if**(elevatorString.size() <2){

**return** **false**;

**break**;

}

**if**(i == 0){

last\_value = elevatorString[i];

**continue**;

}

found = elevatorString.find\_last\_of(ch);

found2 = elevatorString.find\_last\_of(hc);

found = **int**(found);

found2 = **int**(found2);

**int** current\_passengers = 0;

**for**(**int** i = 0; i < elevatorString.size(); i++){

**if**(elevatorString.size() <2){

**return** **false**;

**break**;

}

**if**(i == 0){

last\_value = elevatorString[i];

**continue**;

}

**if**(last\_value == 'M'){

**if**(isdigit(elevatorString[i])){

last\_value = elevatorString[i];

**continue**;

}

**else** **if**(elevatorString[i] != isdigit(elevatorString[i])){

**return** **false**;

**break**;

}

}

While these code blocks seem out of context I have to admit, they are some of the core pieces of code that make my entire elevator project run smoothly. Currently I do not have any test codes that do not seem to work, but I am definitely trying to find them.