First problem I had with the assignment was not being able to assign stuff with the on-guard conditional statements. This was especially important with the CheckX type of states where I check if the the utterance is an acceptable answer (is in the grammar) and assign if uttered stuff (day, hour or boolean) to a variable such as day, time or isFullDay.

After some research I discovered that it is possible to register assign with the 'action' parameter like the following:

```
  CheckDay:{
    entry: {
      type: "spst.speak",
      params: ({ context }) => ({
        utterance: `Alright. ${context.lastResult![0].utterance} ${
          isDayInGrammar(context.lastResult![0].utterance) ? "is a good day." : "is a horrible idea."}`,
      }),
    },
    on: { SPEAK_COMPLETE: [
      { target: "AskIfFullDay",
        guard: ({ context }) => isDayInGrammar(context.lastResult![0].utterance),
        actions: assign(({ context }) => ({ day: getDay(context.lastResult![0].utterance) })), // If the day is in the
grammar, save the day in the context to the day field.
      },
      { target: "AskDay" },
    ] },
  },
```

In this excerpt, for example, we switch to state AskIfFullDay if the utterance is a day in grammar and if so, using the 'actions' parameter, we assign it to the day variable using getDay function we defined earlier. Otherwise, we loop back to the state AskDay in order to repeat the question.

Another major limitation is repeating states. Basically I all of the CheckX states and AskX states look exactly like each other but I had to repeat it. I have found couple of ways to pre-define template states and constantly refer back to them but that was too confusing for me. With this version, my code and states are entirely functional but there is too much repetition.

Another related problem I had was to find a smarter way to define the vocabulary. In my version, I have different entries for each available contact, each time of the day and each

day of the weak. It could have been more elegant to define if there were a way to say like “okay times of the day are basically numbers from 0 to 24” and this list represents the the entire available days etc. However, that required me to use of method called .isIncludes(), and I could not find a way to use it without bumping into errors...