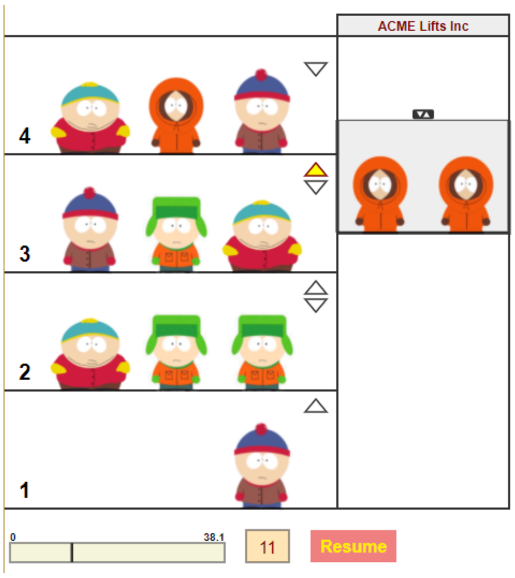
UNSW SYDNEY | ProgComp 2021 **Grand Final**

Task 4. Lift Simulation Visualisation

Available marks: 38



Task 3 in the Main Round 2021 posed an interesting classic problem to sort passengers between floors in an apartment building. The optimal algorithm that minimises the number of lift moves (not passenger moves) was supplied, and most teams solved the problem to some degree.

This task requires you to animate the solution, using the input and correct output for an instance of the problem. To summarise, the first line of input lists three numbers:

- The number of floors, between 1 and 7 inclusive,
- the number of passengers per floor, between 1 and 4 inclusive, and the capacity of the lift, between 1 and 3, inclusive.

Then follows the layout of the building, from the top floor down. Floor 1 is the ground level. Passengers are denoted by the floor number they want to go to, and there's an equal number to go to each floor.

Finally there's the optimal solution for this configuration, in terms of minimum lift moves, not necessarily passenger moves. Each step consists of a line comprising

- The step number. 0 is the initial state, with the empty lift on the ground floor (level 1).
- The direction the lift is about to travel, `UP` or `DN`.
- The floor where the lift is positioned, preceded by an `F`.
- The passengers waiting on the floor. After step 0, it shows the state of the floor after people have moved into and out of the lift, just before the lift is about to move.
- `L:` and the passengers now in the lift.

The image shows the state of the solution for **task4-432a.dat** where the lift has just passed the 3rd floor (step 3) carrying two passengers to their destination on the 4th floor. There they'll be swapped with passengers going to floors 1 and 2 (step 4).

```
3 UP F3 1 2 3 L: 4 4
4 DN F4 4 4 4 L: 1 3
```

Conveniently, the floor and lift contents can be treated as sequences rather than sets, so any passenger who doesn't participate in a move to/from the lift retains their absolute position. This makes it easier to determine who actually moved. You obviously need to track each individual passenger, distinguishing those with the same destination. You can also treat all moves in a step to be simultaneous, with passengers passing front of (or through) each other.

SVG Objects

The file **task4-prolog.svg** contains all the complex SVG objects and some styling values. Your program copies that file and appends the specific SVG elements needed to simulate the solution, and a closing `</svg>` tag. The resulting file must be in the same directory as the the supplied PNG and js files when it's loaded into a browser (Chrome is preferred).

The program will position all required objects, embedding the complete animation code (with `<animate>` specifying timed attribute changes). The animations are mostly movements caused by changing `x` or `y` attributes, except for the lift direction arrows, whose opacity, stroke and fill all need to be varied. Solutions using css animations or other animation schemes are OK as long as they achieve the described effects.

These are the objects you'll need to use.

Object	`xlink:href`	Example of usage	Notes
Passengers	`#P` followed by the passenger number	`<use xlink:href="#P3" id="P3.2" x="100" y="200"> <animate attributeName="x" fill="freeze" begin="12s" dur="2.5s" to="480" /> </use>`	Passenger 3 (`id` suggests it's the second instance), origin is middle of the passenger's feet.
The lift	`#lift` followed by the capacity	`<use xlink:href="#lift2" x="350" y="500"> <animate attributeName="y" fill="freeze" begin="43s" dur="2.5s" to="240" /> </use>`	2-person lift shown, origin is always the bottom left corner.
The lift sign	`#sign` followed by the lift capacity	`<use xlink:href="#sign2" x="400" y="240" />`	Static position at the top of the liftwell. Origin is bottom left of the sign.
The floor separators and liftwell	(lines or paths styled by `floor`)	`<path d="M0,200 h360" class="floor" />`	`path` is slightly easier since all coordinates are embedded in one attribute instead of four.
The floor numbers	(text styled by `floornum`)	`<text x="14.5" y="410" class="floornum">4</text>`	Occupies a fixed position near the left edge.
The floor lift arrows	`#UP` or `#DN`	`<use id="DN6" xlink:href="#DN" x="440" y="185" stroke="#333333" fill="#FFFFFF"> <animate attributeName="stroke" fill="freeze" begin="24.5s" dur="0.1s" to="#800000" /> </use>`	Origin is the middle of the horizontal side of the triangle. Stroke and fill must be literals: Active stroke is `#800000` as shown Active fill is `#FFFF00`
Timing parameters	(params data element)	`<rect id="params" data-max-time="116.1" data-timer-x="100" data-timer-y="700" />`	Dimensionless rectangle passes data to JS. Coords set the top left of the timer group.

Layout and timing

Floors should be 110px apart, waiting passengers should be spaced at 80px centres. Allow an extra 30px horizontal for the lift arrows, which look OK at 85 and 90px above the floor. There should be only an UP arrow on floor 1 and a DN arrow on the top floor. Allow 30px above the top floor for the sign, or more if you want to position the timer there.

The layout should be compactly sized according to the number of floors and queue size, and aligned towards the top left of the image space. This means calculating the liftwell and level 1 floor positions rather than using the same absolute positions for all inputs.

Event timing: try to adhere to these values so the judges can locate certain transitions in time.

- 2s at the start of the animation before the first event.
- 2.5s for people to enter/leave the lift. Each passenger can be moved independently; they don't have to "walk" at the same speed.
- 0.5s after this the lift starts to move.
- 1.5s for the lift to move between floors.
- 0.5 seconds *before the lift arrives* the arrow on the arriving floor flashes: it becomes invisible (opacity=0) over 0.1s, then visible in the active state when the lift arrives, again change opacity over 0.1s. See table for colours.
- When the lift arrives at the new floor the lift arrow on the floor the lift is moving from is inactivated.
- At the end of the animation when the passengers have alighted from the lift on level 1, the UP arrow is inactivated.

The lift must smoothly pass a floor if passengers are not going to enter or leave the lift on that floor. You can flash the arrow on the floor as the lift passes if you like, or suppress it.

Click on the timeline to adjust the current simulated time in either direction for testing and to speed up assessment.

Assessment

The judges will ask you to run your program with some of the many examples, typically one small and one large. You must then show the corresponding animation in a browser. They will ask you to move the simulation times around so the assessment can be done quickly.

The available marks are:

- 14 marks for the initial static state: all objects are displayed in the correct relative positions, and the static elements remain intact over the animation.
- 20 marks for the accuracy of the movements of all passengers and the lift.
- 4 marks for the lift arrow animation. It's separate because the attributes are different. The active and inactive states must be correct as well as the transitions as described above.