### Journey planner

**Instructions for candidates**

Please complete the exercise in one of C#, Java, Kotlin, JavaScript/Typescript or Python and send us a zip of your code, together with the output from running your program at the command line for the two cases below.  
Please don’t share the problem or your solution publicly, e.g. in a public github repository, but feel free to use google drive or similar to send us your code.

If you’re not comfortable in any of the languages above do let us know and we’d be happy to discuss alternatives.

We’re looking for candidates who are able to write clear, robust and maintainable code. We’re not looking for a production-ready solution, but would hope to see something that helps us understand how you would approach a real project. Feel free to use comments to show your thinking or suggestions. Please make sensible assumptions where possible rather than asking clarifying questions.

**The exercise is intended to take under 2 hours so please don’t feel you need to spend longer than that.**We really do appreciate the time investment you’re making here and don’t want to stretch it out. Feel free to send a partial solution if you run out of time.

Softwire have been asked to create a tool for journey planning on a rail network. One of the first requirements is to be able to generate a list of potential routes between two stations.

To calculate routes the tool will be provided with a list of adjacent station pairings in tab-separated format. The file StationsAndLines.tsv has this data for the London Underground network. Each row represents a single link between adjacent stations on a specific line. Some station pairs may be connected by multiple lines (and hence have multiple rows in the file).

**Part 1**

Write a command line tool that takes the names of a starting station and destination station as input parameters and returns a list of valid routes between them (sequences of stations visited). A route is valid if it does not visit any station more than once and is less than 10 minutes slower than the quickest route.

You can assume that journeys take 2 minutes per station, and ignore any time spent changing trains between different underground lines.

Considering the route from Kentish Town to South Wimbledon, you should be able to validate that there are 6 routes using just the Northern and Victoria lines.

**How many valid routes are there from Kentish Town to South Wimbledon using all lines and ignoring transfer times?**

**Part 2**

Update your program to account for the need to change trains. Assume that any time you need to change to a different line adds 5 minutes to your journey. Ignore changes to different branches of the same line.

As before, your program should return the list of valid routes. Extend this to also output (for each route) the journey time, and details of the changes you need to make along the route (i.e. which station(s) you need to change at, and to which line(s)).

**How many valid routes are there for the journey from Kentish Town to Wimbledon Park, taking into account time spent changing trains?**   
*(Note this is a different end station and slightly more complicated journey than part 1)*