**TESTING:**

Testing Table – Section 17:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Objective Number** | **Test Number** | **Test Description** | **Data Values** | **Data Type** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Improvement Needed** |
| 1, 1.2, 2, 2.1, 2.2 | 1a | Test if the correct route will be outputted based on what order the 2 inputs are inputted. | Image 1 | Normal | A valid route is output. | Image 2 | Pass | None |
| 1, 1.2, 2, 2.1, 2.2 | 1b | Test if the correct route will be outputted based on what order the 2 inputs are inputted. | Image 3 | Normal | The same route from 1a is outputted, but in reverse order. | Image 4 | Fail – logical error | Fix Dijkstra’s algorithm so that the best route is output no matter which order the inputs are inputted.  Image 5 is code before the changes. |
| 9.2 | 2 | Test if the data from the TfL API is outputted correctly. | Image 6 | Normal | Data from the TfL API on status updates of the correct line is outputted | Image 7 | Fail | The endpoint used should be changed to an endpoint which is valid. |
| 9.2 | 3 | Test if the data from the TfL API is accessed. | n/a | n/a | Data from the TfL API is accessed and printed in the console. It should not be specific for a line yet. | Image 8 | Pass | None |
| 9.2 | 4a | Test if the data from the TfL API is outputted correctly after successfully accessing it and the html for outputting an image corresponding to the input given. | Image 9 | Normal | Data from the API is outputted for the specific line given in the input and the html for an image of the line is also outputted. | Image 10 | Pass | None |
| 9.2 | 4b | Test if the data from the TfL API is outputted correctly after successfully accessing it and the html for outputting an image corresponding to the input given is correct. | Image 11 | Erroneous | Data from the API is not outputted as the data inputted is not valid. | Image 12 | Pass | None |
| 1, 1.2, 1.3, 1.4, 2, 2.1, 2.2, 8.1 | 5a | Test if the shortest route has been outputted with normal data values. | Image 13 | Normal | The shortest route between the two stations is outputted which should be:  “Paddington,  Bayswater,  Notting Hill Gate,  Queensway,  Lancaster Gate,  Marble Arch”  “740.1 seconds” | Image 14 | Fail | Dijkstra’s algorithm has a problem, so change the function by changing the way in which the algorithm checks if the user would have to change lines. If there is a change of line, then there must be an added 500 seconds to account for this.  At the moment, the algorithm tries to do this by looking through a 2D array for the two stations that are being traversed. This does not work as there can be multiple lines used for the same two stations, and it is hard to record which line was used previously if there is a nested for loop in order to travel to all nodes. The lines used would have to be stored somewhere, but an improvement can be made so that the changeover time is included within the adjacency dictionary.  Image 15 shows the way in which the program tries to determine if the route requires a line change. |
| 1, 1.2, 1.3, 1.4, 2, 2.1, 2.2, 8.1 | 5b | Test if the shortest route has been outputted with normal data values. | Image 13 | Normal | The shortest route between the two stations is outputted which should be:  “Paddington,  Bayswater,  Notting Hill Gate,  Queensway,  Lancaster Gate,  Marble Arch”  “740.1 seconds” | Image 16 | Pass | None (code before improvement and after improvement since test 5a is shown in image 17). |
| 1, 1.2, 1.3, 1.4, 2, 2.1, 2.2, 8.1 | 5c | Test if the route is the same regardless of which order the 2 stations are inputted. | Image 18 | Normal | The shortest route between the two stations is outputted which should be:  “Marble Arch,  Lancaster Gate, Queensway, Notting Hill Gate, Bayswater, Paddington”  “740.1 seconds” | Image 19 | Pass | None (Dijkstra’s algorithm function works correctly). |
| 1, 1.2, 1.3, 1.4, 2, 2.1, 2.2, 8.1, 8.2 | 6a | Test if a station including the line name (such as “Piccadilly Circus” since “Piccadilly” is a line name as well) is a valid input for the journey planner. | Image 20 | Boundary | The shortest route between the two stations will be outputted which should be:  “Piccadilly Circus,  Oxford Circus”  […] | Image 21 | Fail | Change any station names which may result in an error. |
| 1, 1.2, 1.3, 1.4, 2, 2.1, 2.2, 8.1, 8.2 | 6b | Test if a station including the line name (such as “Piccadilly Circus” since “Piccadilly” is a line name as well) is a valid input for the journey planner. | Image 20 | Boundary/ Normal | The shortest route between the two stations will be outputted which should be:  “Piccadilly Circus,  Oxford Circus”  […] | Image 22 | Pass | None (code with improvements is image 23). |
| 5, 5.1, 5.2, 5.3 | 7a | Test to see if the timings will be multiplied when “rush\_hour” and “weekend” variables are both “Yes”. | Image 24 | Normal | The timings for the route should be outputted as 248.75 seconds since the timings should be multiplied by 2.5. | Image 25 | Pass | Program works as expected, but it is not very efficient at the moment, so it can be improved by making the code more concise and using less repeated code. |
| 5, 5.1, 5.2, 5.3 | 7b | Test to see if the timings will be multiplied when “rush\_hour” and “weekend” variables are both “Yes”. The same as test 7a, but the difference is that the code is improved to be more efficient. | Image 26 | Normal | The timings for the route should be outputted as 248.75 seconds since the timings should be multiplied by 2.5. | Image 27 | Pass | Program was improved after test 7a. Code before improvements is shown in image 28 and code after improvements is shown in image 29. |
| 1.1, 4.3, 7, 7.1 | 8a | Test to see what happens when a non-step free station is inputted as one of the stations and the user has requested a step-free route. | Image 30 | Erroneous | A route will not be outputted. | Image 31 | Pass – should have an error message. | Make an error message display when the route is step free but the stations are not. |
| 1.1, 4.3, 7, 7.1 | 8b | Test to see what happens when a non-step free station is inputted as one of the stations and the user has requested a step-free route. | Image 30 | Boundary | An error message will be displayed. | Image 32 | Fail | Program was improved after test 8a. Code before improvements is shown in image 33 and 34, code after improvements is shown in image 35 and image 36. |
| 9, 9.1 | 9a | Test to see if the file “arrivaltimes.pyw” works correctly by outputting a message when a station has no arrivals. | Image 37 | Normal | A relevant message will be output:  “No arrivals at this station at the moment, sorry” | Image 38 | Pass | None (code is shown in image 39). |
| 9, 9.1 | 9b | Test to see if the file “arrivaltimes.pyw” works correctly by outputting relevant arrival times. | Image 40 | Normal | The arrivals will be displayed in order they were received, with information about the time and platforms for each arrival. | Image 41 | Pass | None (code is shown in image 42). |
| 9, 9.2 | 10a | Test to see if the file “status\_updates.pyw” works correctly by outputting a message when a line has no disruptions. | Image 43 | Normal | The severity description will be displayed about the line given. The message outputted will be “No disruptions”. | Image 44 | Pass | None (code is shown in image 45). |
| 9, 9.2 | 10b | Test to see if the file “status\_updates.pyw” works correctly by outputting a message when a line has minor delays. | Image 46 | Normal | The severity description will be displayed about the line given. The message outputted will be “Minor Delays”. | Image 47 | Pass | None (code is shown in image 48). |
| 9, 9.2 | 10c | Test to see if the file “status\_updates.pyw” works correctly by outputting a message when a line has severe delays. | Image 49 | Normal | The severity description will be displayed about the line given. The message outputted will be “Severe Delays”. | Image 50 | Pass | None (code is shown in image 48). |
| 9 | 11a | Test to see if the file “liftdisruptions.pyw” works correctly by outputting a message when a station has no lift disruptions. | Image 51 | Normal | A message will outputted which says that there are no lift disruptions at this station at the moment. | Image 52 | Pass | None (code is shown in image 53). |
| 9 | 11b | Test to see if the file “liftdisruptions.pyw” works correctly by outputting a message when a station has a lift disruption. | Image54 | Normal | Information about the lift disruption will be outputted. | Image 55 | Pass | None (code is shown in image 53). |
| 1.1, 4.3, 7, 7.1 | 12 | Test to see if the step-free journey planner works correctly by inputting valid values. | Image 56 | Normal | The shortest route between the two step-free stations will be outputted which should be:  “Oxford Circus”  “Green Park”  “Bond Street”  […] | Image 57 | Pass | None. |
| 1, 1.5, 8, 8.2 | 13a | Test if route is output using normal journey planner in correct format on web page. | Image 58 | Normal | The shortest route between the two step-free stations will be outputted on the website which should be:  “Leicester Square”  “Tottenham Court Road”  “Oxford Circus”  […] | Image 59 | Pass | None. |
| 1, 1.5, 8, 8.2 | 13b | Test if step-free route is output in the correct format on the webpage. | Image 60 | Normal | The shortest route between the two step-free stations will be outputted on the website which should be:  “Oxford Circus”  “Green Park”  “Bond Street”  […] | Image 61 | Pass | None. |
| 3.3, 9, 9.1 | 14a | Test if arrival times are output in the correct format on the webpage. | Image 62 | Normal | The arrivals will be displayed on the website in order they were received, with information about the time and platforms for each arrival. | Image 63 | Pass | None. |
| 3.3, 9, 9.2 | 14b | Test if status updates are output in the correct format on the webpage. | Image 64 | Normal | The status of the disruptions at the given station will be outputted and whether there is good, or poor service will also be outputted. An image of the line entered will be displayed. | Image 65 | Pass | None. |
| 3.3, 9 | 14c | Test if lift updates are output in the correct format on the webpage. | Image 66 | Normal | Information on any lift disruptions will be displayed on the website with information on why the lift is not in use and its unique lift ID. | Image 67 | Pass | None. |
| 10, 10.1 | 15 | Test if the web server works in order to provide the web page as a resource for a client. | n/a | n/a | When using the link:  <http://localhost:8080/index.html> on the computer that the webserver.pyw file is run on, the about page should appear. The user should be able to navigate the whole website through this page by using the navbar. | Image 68 | Pass | None. |
| 1.1, 1.5, 3, 3.2, 4, 4.1, 4.2, 4.3, 7, 7.1, 7.2 | 16a | Test if the form for the journey planner is output in the correct format on the webpage. | n/a | n/a | Text asking the user for the start station and end station will appear on the webpage as well as the header, footer, and the navbar. There will be 2 textboxes for these inputs. There will be “Yes”/”No” questions for Rush Hour, the Weekend, and if the route is Step-Free. | Image 69 | Pass | None. |
| 3, 3.2, 4, 4.1, 4.2, 7, 7.1, 7.2 | 16b | Test if the form for arrival times is output in the correct format on the webpage. | n/a | n/a | Text asking the user for a station name and a textbox will appear on the webpage as well as the header, footer, and the navbar. | Image 70 | Pass | None. |
| 3, 3.1, 3.2, 4, 4.1, 4.2, 7, 7.1, 7.2 | 16c | Test if the form for status updates is output in the correct format on the webpage. | n/a | n/a | Text asking the user for a line name and a textbox will appear on the webpage as well as the header, footer, and the navbar. | Image 71 | Pass | None. |
| 3, 3.1, 3.2, 4, 4.1, 4.2, 7, 7.1, 7.2 | 16d | Test if the form for lift updates is output in the correct format on the webpage. | n/a | n/a | Text asking the user for a station name and a textbox will appear on the webpage as well as the header, footer, and the navbar. | Image 72 | Pass | None. |
| 3, 3.1, 3.2, 4, 4.1, 4.2, 6, 6.1, 6.2 | 16e | Test if the html for “index.html” is output in the correct format on the webpage. | n/a | n/a | Information on the program and an image will appear on the webpage as well as the header, footer, and the navbar. | Image 73 | Pass | None. |
| 3, 3.1, 3.2, 4, 4.1, 4.2 | 16f | Test if the html for “map.html” is output in the correct format on the webpage. | n/a | n/a | The image of the tube map will appear on the webpage as well as the header, footer, and the navbar. | Image 74 | Pass | None. |

Images Referenced Table – Section 18:

|  |  |  |  |
| --- | --- | --- | --- |
| **Image Number:** | **Test Number:** | **Image:** | **Explanation:** |
| 1 | 1a | Text  Description automatically generated | Inputted normal data values. “Paddington” is station1 and “Marble Arch” is station2, but in test 1b, these are swapped around to see if they will output the same route (as Dijkstra’s algorithm should output the same route regardless of which order the stations have been inputted). |
| 2 | 1a | A computer code with blue text  Description automatically generated | A valid route is outputted, however this may not be the best route, so the test has passed, but the program could be changed to give a better, faster route. |
| 3 | 1b | A screenshot of a computer  Description automatically generated with low confidence | Same inputs as test 1a but the stations have been swapped around to see the effect of changing the order in which stations are inputted. |
| 4 | 1b | Graphical user interface, text, application  Description automatically generated | Again, a valid route has been outputted, but this is not the best route. Additionally, this does not match the route from test 1a, therefore, there is a problem with the Dijkstra’s algorithm function which needs to be fixed. |
| 5 | 1b 1a | A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generatedFunction which is called: | This is the improvement for test 1a and 1b because, although test 1a had passed, test 1b did not output the same route as test 1a, therefore there is a problem with both tests. After doing some calculations, the best route that should be output for that test is neither of the routes that have been outputted. This image shows the Dijkstra’s algorithm function (specifically the lines of code which are causing the problem). |
| 6 | 2 | Icon  Description automatically generated with medium confidence | Tested any line that is a valid input. |
| 7 | 2 | Text  Description automatically generated | An error occurs when the program runs. The reason for this is the endpoint not being valid for the TfL API. |
| 8 | 3 | Text  Description automatically generated | Information has been accessed from the TfL API and is outputted. However, the output needs to be improved so that it is only the relevant information that the user has requested. |
| 9 | 4a |  | Valid input given (normal data). “line” is set as “Central” so that the information on the status updates of the central line can be outputted. |
| 10 | 4a | Text  Description automatically generated | The program correctly outputs the information from the API which is regarding the line inputted by the user. |
| 11 | 4b | A picture containing logo  Description automatically generated | Invalid input given (erroneous data). |
| 12 | 4b | Text  Description automatically generatedText  Description automatically generated  The information from the API would be outputted here, but nothing has been outputted, so the data has not been accessed. | The program crashes if the erroneous data has been inputted. The API does not return any data once the input is invalid, but when choosing an image based on which line was chosen, the program crashes.  This is acceptable, as the user will have to pick between lines from a drop down menu to ensure that a valid input has been given. |
| 13 | 5a, 5b | Text  Description automatically generated | Normal data values inputted. |
| 14 | 5a | Text  Description automatically generated | The actual output is not the expected output. The program has given a route which is valid but, this is not the quickest route. Therefore, Dijkstra’s algorithm does not work as intended. This route is 1021.8 seconds, whereas another route is 740.1 seconds, and this should have been the route which was outputted. |
| 15 | 5a | A screenshot of a computer  Description automatically generatedText  Description automatically generated | Images of the code trying to determine if there has been a line change and if so, then add a changeover time. This method does not work, so improvements need to be made. The code tries to determine which line is used by looking through a 2D array containing all of the valid routes. The problem is that multiple lines can be used, and it is hard to store values for which line was previously used to get to the node that is being traversed from. |
| 16 | 5b | Text  Description automatically generated | Actual output is the expected output, indicating that Dijkstra’s algorithm works after the new improvement made. |
| 17 | 5b | A screenshot of a computer  Description automatically generatedSnippet of stations\_adjacency\_dict before improvements:  Text, letter  Description automatically generatedSnippet of stations\_adjacency\_dict after improvements:  Each station has changed to have multiple “substations”. These are used to determine which line has been used, as the values are dependent on the line. For example, from Notting Hill Gate, you can reach Queensway. This is shown in the stations\_adjacency\_dict before the improvements. However, the difference in the new stations\_adjacency\_dict is that you can reach the central line platform at Queensway if you leave from the central line platform at Notting Hill Gate. Additionally, you can travel between the platforms at each station and this takes 5 minutes for each changeover. | The adjacency dictionary before and after improvements (before the test failed, after passed) are shown. After changing the adjacency dictionary to be like this, the program no longer needs to check if the route requires a change of line, since it is included in the adjacency dictionary. Additionally, this reverts Dijkstra’s algorithm back to the 4 key steps and removes the “added” step of checking for lines (which made it not work logically). |
| 18 | 5c | Text  Description automatically generated | Normal data values inputted. |
| 19 | 5c | Text, letter  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 20 | 6a,6b | Text  Description automatically generated | station1 is “Piccadilly Circus”. “Piccadilly” is also the name of a line, so this test is to see whether this affects the route. |
| 21 | 6a | Text  Description automatically generated | An error occurs because of the station name. Therefore, improvement is needed to change the station name so that it does not interfere with the “dijkstra” function. |
| 22 | 6b | Text, letter  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 23 | 6b | Graphical user interface, application  Description automatically generated | Changes made after test 6a are shown in this image. |
| 24 | 7a | Text  Description automatically generated | Same inputs as test 6a, but the “rush\_hour” and “weekend” variables are set to “Yes” to test if timings are changed. |
| 25 | 7a | Text  Description automatically generated | Actual output is the expected output, so the code has passed this test. However, the code if very inefficient, using nested for loops and rewritten code, so I decided to re-write the code so that it is more concise. |
| 26 | 7b | Graphical user interface, text  Description automatically generated | Same inputs as test 7a, but the “rush\_hour” and “weekend” variables are set to “True” to test if timings are changed when the code has been updated. |
| 27 | 7b | Text  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 28 | 7b | Graphical user interface, text, application  Description automatically generated | Code works as required, but lots of repeated code which could be changed. Improvements are shown in the next image. |
| 29 | 7b | A picture containing graphical user interface  Description automatically generated | Code from previous image is improved by making code more concise and efficient. |
| 30 | 8a, 8b | Graphical user interface, text  Description automatically generated | “station2” is assigned as “Pimlico”, but it is not a station with step-free access. Therefore, this test is to see what happens when a non-step free station is inputted with “step\_free” as “Yes”. |
| 31 | 8a | Text, letter  Description automatically generated | No route is outputted because the stations are invalid for the step-free journey planner. |
| 32 | 8b | Text  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 33 | 8b | Graphical user interface, text, application  Description automatically generated with medium confidence | Code works as required, but an error message could be displayed when a step free route is input with a non-step free station. Improvements are shown in image 35. |
| 34 | 8b | A picture containing text  Description automatically generated | HTML file showing the drop-down menu of all of the stations which the user can input. This has been improved to show the user which stations are step free (shown in image 36). |
| 35 | 8b | Graphical user interface, text, application  Description automatically generated | Else statement is added for when the start and end stations are invalid so that an error message is displayed. |
| 36 | 8b | Table  Description automatically generated with low confidence | HTML file shows dropdown menu of the stations for the journey planner. Every station which can be used in the step free journey planner is changed so that it has (Step-free) written next to it. |
| 37 | 9a |  | “Victoria” is assigned to the variable called “station” to see the arrivals at this station. |
| 38 | 9a | Text  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 39 | 9a | Graphical user interface, text, application  Description automatically generated | Code is shown from test 9a. |
| 40 | 9b |  | “Paddington” is assigned to the variable called “station” to see the arrivals at this station. |
| 41 | 9b | Text  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 42 | 9b | A picture containing table  Description automatically generated | Code is shown from test 9b. |
| 43 | 10a |  | “Victoria” is assigned to the variable called “line” to see the status updates of the Victoria line. |
| 44 | 10a | Graphical user interface, text, application  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 45 | 10a | Text  Description automatically generated | Code is shown from test 10a. |
| 46 | 10b | Icon  Description automatically generated with low confidence | “District” is assigned to the variable called “line” to see the status updates of the District line. |
| 47 | 10b | Graphical user interface, text, application  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 48 | 10b, 10c | Text  Description automatically generated | Code is shown from test 10b and 10c. |
| 49 | 10c | Icon  Description automatically generated with medium confidence | “Central” is assigned to the variable called “line” to see the status updates of the Central line. |
| 50 | 10c | Graphical user interface, text, application  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 51 | 11a | A screenshot of a computer  Description automatically generated | “Victoria” is assigned to the variable called “station” to see the lift disruptions at this station. |
| 52 | 11a | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 53 | 11a, 11b | A screenshot of a computer  Description automatically generated | Code is shown from test 11a and 11b. |
| 54 | 11b | A screenshot of a computer  Description automatically generated | “Tottenham Court Road” is assigned to the variable called “station” to see the lift disruptions at this station. |
| 55 | 11b | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 56 | 12 | A screenshot of a computer  Description automatically generated | “Oxford Circus” is assigned to the variable called “station1”, “Bond Street” is assigned to the variable called “station2”, “No” is assigned to “rush\_hour” and “weekend”. “Yes” is assigned to the variable called “step\_free”. |
| 57 | 12 | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 58 | 13a | A screenshot of a computer  Description automatically generated | Start station is entered as “Leicester Square” and the End station is entered as “Oxford Circus”. The Rush Hour, Weekend, and Step Free values are set as “No”. |
| 59 | 13a | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 60 | 13b | A screenshot of a computer  Description automatically generated | Start station is entered as “Oxford Circus” and the End station is entered as “Bond Street”. The Rush Hour, Weekend, are set as “No” and Step Free is set as “Yes”. |
| 61 | 13b | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 62 | 14a | A screenshot of a computer  Description automatically generated | Station is entered as “Bank” to see the arrivals at this station. |
| 63 | 14a | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 64 | 14b | Graphical user interface, text, website  Description automatically generated | Line is entered as “central” to see the status updates of this line. |
| 65 | 14b | Graphical user interface  Description automatically generated with medium confidence | Actual output is the expected output, so the code has passed this test. |
| 66 | 14c | A screenshot of a computer  Description automatically generated | Station is entered as “Bank” to see the lift updates at this station. |
| 67 | 14c | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 68 | 15 | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 69 | 16a | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 70 | 16b | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 71 | 16c | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 72 | 16d | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 73 | 16e | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |
| 74 | 16f | A screenshot of a computer  Description automatically generated | Actual output is the expected output, so the code has passed this test. |