AI Assignment 2 Output

Aryan GD Singh
2019459

After consulting the main.pl file, we start our program by calling the start predicate

```
?- start.
readrow(row(ahmedabad,bangalore,bhubaneshwar,bombay,calcutta,chandigarh,cochin,delhi,hyderabad,indore,jaipur,kanpur,lucknow,madras,nagpur,nasik,panjim,patn
a,pondicherry,pune))
readrow(row(agartala,3305,3824,2286,3593,1863,2998,4304,2708,3330,2891,2801,2281,2252,3493,2696,3365,3507,1681,3661,3442))
readrow(row(agra,878,1848,1578,1202,1300,448,2278,200,1246,591,230,290,369,2048,770,1005,1715,885,2210,1214))
readrow(row(ahmedabad,-,1490,1697,552,2068,1157,1845,911,1436,442,648,1168,1247,1821,965,504,1165,1656,1818,664))
readrow(row(allahabad,1251,1686,1090,1457,817,912,2216,650,1084,803,713,193,234,2011,608,1155,1419,402,1077,1364))
readrow(row(amritsar,1356,2496,2224,1849,1919,239,3163,445,1892,1258,706,926,939,2688,1416,1665,2237,1531,2856,1862))
readrow(row(asansol,1842,2187,523,2040,226,1503,2544,1262,1693,1394,1304,789,825,1857,1122,1746,2300,395,2024,1955))
readrow(row(bangalore,1490,-,1538,1013,1961,2296,512,2049,563,1601,2005,1855,1934,331,1078,1035,440,2071,328,826))
readrow(row(baroda,119,1408,1604,433,1937,1181,1763,1151,1127,379,789,1230,1311,1739,774,457,1158,1582,1735,545))
readrow(row(bhopal,523,1148,1162,778,1495,988,1995,742,808,191,596,585,664,1604,352,605,1143,1016,1772,814))
readrow(row(bhubaneshwar,1647,1538,-,1679,423,2026,1895,1713,1044,1355,1758,1283,1254,1207,830,1516,1455,862,1375,1587))
readrow(row(bombay,552,1013,1678,-,2012,1645,1368,1404,729,589,1148,1278,1366,1344,849,197,584,1856,1452,184))
readrow(row(calcutta,2068,1461,423,2012,-,1721,2318,1474,1467,1620,1530,1010,1089,1160,1163,1849,974,621,1798,2058))
readrow(row(calicut,1648,520,1923,1171,2346,2741,222,2494,910,1998,2523,2260,2339,715,1483,1193,576,2476,566,984))
readrow(row(chandigarh,1157,2296,2026,1645,1721,-,1965,248,1693,1052,507,661,740,2489,1217,1466,2028,1332,2657,1663))
readrow(row(cochin,1845,512,1895,1368,2318,1965,-,2718,1090,1804,2745,2385,2572,687,1608,1390,798,2601,530,1181))
readrow(row(coimbatore,1669,333,1633,1192,2057,2669,195,2412,912,1964,2369,2218,2297,426,1441,1214,800,2434,410,1005))
readrow(row(delhi,911,2049,1713,1404,1474,248,2718,-,1477,806,263,481,499,2243,971,1220,1782,1086,2411,1417))
readrow(row(gwalior,919,1734,1486,1085,1224,462,1881,315,1132,497,351,280,359,1928,656,891,1467,809,2096,1100))
readrow(row(hubli,1101,391,1620,614,2032,2101,774,1854,486,1060,1060,1772,1851,683,995,646,190,1998,653,437))
readrow(row(hyderabad,1436,563,1044,729,1467,1693,1090,1447,-,999,1404,1253,1332,699,476,754,765,1469,865,545))
readrow(row(imphal,3240,3677,2139,3446,1717,2851,4156,2561,3283,2744,2654,2134,2105,3326,2549,3218,3360,1534,3614,3295))
readrow(row(indore,442,1601,1355,589,1620,1052,1804,806,999,-,405,689,768,1795,445,414,1115,1205,1963,623))
readrow(row(jabalpur,901,1335,1087,1143,1167,1046,1885,800,733,859,845,543,584,1529,257,943,1088,736,1697,1003))
readrow(row(jaipur,648,2005,1758,1148,1530,507,2745,263,1404,405,-,517,598,2200,928,1248,2496,1115,2368,1371))
readrow(row(jamshedpur,1225,1690,422,1916,268,1602,2710,1356,1578,1477,1387,867,1121,1629,1102,1788,2229,473,1797,1997))
readrow(row(jullundur,1285,2416,2413,1778,1869,154,3082,375,1821,1171,891,855,868,2617,1354,1591,2146,1460,2785,1791))
readrow(row(kanpur,1168,1855,1283,1278,1010,661,2385,481,1253,689,517,-,79,2049,777,1103,1813,596,2217,1312))
readrow(row(kolhapur,911,484,1622,426,2045,1910,934,1664,578,870,1518,1779,1858,910,1050,456,254,2047,907,247))
readrow(row(lucknow,1247,1934,1254,1366,1089,740,2572,499,1332,768,598,79,-,2128,856,1182,1883,566,2296,1391))
readrow(row(ludhiana,1220,2358,2088,1770,1783,105,3027,310,1756,1115,570,790,803,2552,1280,1528,2091,1395,2720,1726))
readrow(row(madras,1821,331,1207,1344,1630,2489,687,2243,699,1795,2200,2049,2128,-,1272,1366,909,2096,168,1157))
readrow(row(madurai,1922,432,1687,1458,2110,2785,326,2539,995,2091,2496,2345,2424,480,1568,1467,872,2561,333,1238))
readrow(row(meerut,1092,2072,1822,1468,1497,381,2741,66,1470,815,327,537,453,2266,994,1242,1805,934,2434,1440))
readrow(row(nagpur,965,1078,830,849,1163,1217,1608,971,476,445,928,777,856,1272,-,700,1247,993,1440,909))
readrow(row(nasik,504,1036,1516,197,1849,1466,1390,1220,754,414,1246,1103,1182,1366,700,-,701,1679,1363,209))
readrow(row(panjim,1165,440,1455,584,974,2028,798,1782,765,1115,2496,1813,1883,909,1247,701,-,1804,739,501))
readrow(row(patna,1656,2071,862,1856,621,1332,2601,1086,1489,1205,1115,595,566,2096,993,1679,1804,-,2264,1738))
readrow(row(pondicherry,1818,328,1375,1452,1798,2657,530,2411,867,1963,2368,2217,2296,168,1440,1363,739,2264,-,1154))
readrow(row(pune,664,826,1587,184,2058,1663,1181,1417,545,623,1371,1312,1391,1157,909,209,501,1738,1154,-))
readrow(row(ranchi,1781,2098,560,1816,414,1480,2455,1214,1434,1333,1243,723,912,1767,958,1615,1630,302,1935,1675))
readrow(row(shillong,2839,3242,1640,3011,1281,2416,3599,2126,1309,2309,2229,1699,1670,2911,2114,2800,2925,1019,3079,2860))
readrow(row(shimla,1256,2395,2127,1753,1839,99,3056,365,1812,1171,636,846,659,2588,1336,1567,2130,1434,2759,1782))
readrow(row(surat,263,1264,1579,301,1912,1325,1593,1182,983,607,932,1296,1375,1695,749,262,913,1742,1789,412))
readrow(row(trivandrum,2197,696,1953,1376,2376,3051,218,1950,1261,2156,2782,2611,2690,748,1834,1742,792,2827,622,1533))
readrow(row(varanasi,1373,1791,968,1597,695,1032,2331,745,1189,925,835,315,286,1985,713,1399,1524,280,2153,1459))
readrow(row(vijayawada,1705,637,775,973,1198,1959,1174,1716,269,1255,1660,1519,1598,432,742,1428,876,1664,598,842))
readrow(row(vishakapatnam,1815,1093,445,1754,868,2127,1449,1881,599,1433,1838,1687,1765,762,910,1596,1206,1334,930,1658))
```

This reads the csv file and creates predicates for distance

Now we can choose whether to perform Depth first search or Best first search

```
Choose the search method
1. Depth First Search
2. Best First Search
|: 45.
Enter 1 or 2
true.

?-
```

If we enter any value other than 1 or 2 we get the above message.

Depth First Search

We input the start and end cities and get back the path and path cost.

```
Choose the search method
1. Depth First Search
2. Best First Search
|: 1.
Enter Start city: |: agra.
Enter End city: |: shimla.

Path: [agra,ahmedabad,shimla]
Cost: 2134
true .
```

If the same cities are input then we get the following result.

```
Choose the search method
1. Depth First Search
2. Best First Search
|: 1.
Enter Start city: |: agra.
Enter End city: |: agra.

Same start and end point
Path: []
Cost: 0
true.
```

Best First Search

Heuristic:

The heuristic function is defined as follows -

For end city C:

h(A) = min(distance(A, B) + distance(B, C))
where B can be any city, including A and C.

For this heuristic to be admissible and consistent, one assumption has to be made, that the minimum distance between any 2 cities is given by either the direct road b/w them or a path going through maximum 3 cities.

We input the start and end cities and get back the path and path cost.

```
Choose the search method
1. Depth First Search
2. Best First Search
|: 2.
Enter Start city: |: agra.
Enter End city: |: shimla.

Path: [agra,chandigarh,shimla]
Cost: 547
true .
```

We can see that Best first search returns a lower cost path than Depth first search for the same input.

If the same cities are input then we get the following result.

```
Choose the search method
1. Depth First Search
2. Best First Search
|: 2.
Enter Start city: |: imphal.
Enter End city: |: imphal.

Same start and end point
Path: []
Cost: 0
true.
```