

```
1 //Fatima Ansari
2 //BCSF20A046
3
4 #include <iostream>
5 #include <fstream>
6 #include <cmath>
7 using namespace std;
8
9 void validate(unsigned long long int&, int, unsigned long long int);
10 void storeInputToFile(ofstream&);
11 int main()
12 {
13     ifstream fin;
14     ofstream fout;
15     int R, C, F, N, B, T;
16     int a, b, x, y, s, f;
17     fout.open("inputFile.txt");
18     if (fout)
19     {
20         storeInputToFile(fout);
21         fout.close();
22     }
23     else
24     {
25         cout << "ERROR!\n";
26         cout << "File not Created Successfully!\n";
27         exit(0);
28     }
29     fin.open("inputFile.txt");
30     if (!fin)
31     {
32         cout << "ERROR!!\n";
33         cout << "The input File is not found\n";
34         exit(0);
35     }
36     else
37     {
38         // "\nNow, As we Know that we already stored the valid data to the
39         // "so, no need of validation from getting data back from the
40         fin >> R >> C >> F >> N >> B >> T;
41         // Now to assign rides to the vehicles
42         int vehicle = 0;
43         int score = 0;
44         int rides = 0;
45         int simulation = 0;
46         fout.open("outputFile.txt");
47         if (!fout)
```

```
48     {
49         cout << "Output File not Created Successfully\n";
50         exit(0);
51     }
52     else
53     {
54         bool finalStep = false; // to check for final step simulation
55         while (rides < N && !finalStep)
56         {
57             int previousRides = rides; // to store the previous rides
58             if ((N - previousRides) / (F - vehicle))
59             {
60                 rides += (N - previousRides) / (F - vehicle);
61                 if ((N - previousRides) % (F - vehicle++))
62                     rides++;
63             }
64             else
65             {
66                 rides += N - previousRides;
67             }
68             fout << rides - previousRides << " "; // No of rides to each vehicle ↗
69             int i = 0, j = 0; // current position of each vehicle ↗
70             int k = previousRides;
71             while (k < rides && !finalStep)
72             {
73                 simulation = 0;
74                 fin >> a >> b >> x >> y >> s >> f;
75                 if (i == a && j == b) // check for bonus
76                     score += B; // to increase the total score ↗
77                 // Now to go to the
78                 if (i < a)
79                     while (i < a)
80                     {
81                         i++;
82                         simulation++;
83                     }
84                 else
85                     while (i > a)
86                     {
87                         i--;
88                         simulation++;
89                     }
90                 if (j < b)
91                     while (j < b)
92                     {
93                         j++;
```

```
94         simulation++;
95     }
96     else
97         while (j > b)
98         {
99             j--;
100             simulation++;
101         }
102     // Now to handle the earliest start
103     while (simulation < s)
104         simulation++;
105     // Now the vehicle is on the starting position
106     // so, it for destination
107     if (i < x)
108         while (i < x)
109         {
110             i++;
111             simulation++;
112         }
113     else
114         while (i > x)
115         {
116             i--;
117             simulation++;
118         }
119     if (j < y)
120         while (j < y)
121         {
122             j++;
123             simulation++;
124         }
125     else
126         while (j > y)
127         {
128             j--;
129             simulation++;
130         }
131     // now to check for score and latest finish
132     if (simulation < f)
133         score += abs(x - a) + abs(y - b);
134     else if (simulation >= T)
135         finalStep = true;
136     fout << k << " ";
137     k++;
138 }
139 fout << endl;
140 }
141 fout.close();
142 }
```

```
143     cout << "\nTotal Score = " << score << endl
144         << endl;
145     fin.close();
146 }
147
148 return 0;
149 }
150 void validate(unsigned long long int& val, int r1, unsigned long long int r2)
151 {
152     while (val < r1 || val > r2)
153     {
154         cout << "ERROR!! Invalid input\nRe-enter in range of (" << r1 <<
155             "-" << r2 << "): ";
156         cin >> val;
157     }
158 }
159 void storeInputToFile(ofstream& fout)
160 {
161     unsigned long long int R, C, F, N, B, T, temp;
162     cout << "Enter number of row of the gird (1 <= R <= 10000): ";
163     cin >> R;
164     validate(R, 1, 10000);
165     fout << R << " ";
166     cout << "Enter number of columns of the gird (1 <= C <= 10000): ";
167     cin >> C;
168     validate(C, 1, 10000);
169     fout << C << " ";
170     cout << "Enter number of vehicles in the fleet (1 <= F <= 1000): ";
171     cin >> F;
172     validate(F, 1, 1000);
173     fout << F << " ";
174     cout << "Enter number of rides (1 <= N <= 10000): ";
175     cin >> N;
176     validate(N, 1, 10000);
177     fout << N << " ";
178     cout << "Enter Pre-ride bonus for starting the ride on time (1 <= B <=
179         10000): ";
180     cin >> B;
181     validate(B, 1, 10000);
182     fout << B << " ";
183     cout << "Enter number of steps in the simulation (1 <= T <= 10^9): ";
184     cin >> T;
185     validate(T, 1, pow(10, 9));
186     fout << T << "\n";
187     cout << "The first line of the output file has been successfully
188         stored to the file\n";
189     // Now going to the next Step
190     // Taking input for each rides
```

```
188     cout << "\nNow to take more info about each ride\n\n";
189     for (int i = 0; i < N; i++)
190     {
191         // Getting input for ride i
192         cout << "\tRide #" << i << " information\n";
193         cout << "Enter row of the start intersection (0 <= a <= " << R << "
194             << "): ";
195         cin >> temp;
196         validate(temp, 0, R);
197         fout << temp << " ";
198         cout << "Enter column of the start intersection (0 <= b <= " << C << "
199             << "): ";
200         cin >> temp;
201         validate(temp, 0, C);
202         fout << temp << " ";
203         cout << "Enter row of the finish intersection (0 <= x <= " << R << "
204             << "): ";
205         cin >> temp;
206         validate(temp, 0, R);
207         fout << temp << " ";
208         cout << "Enter column of the finish intersection (0 <= y <= " << C << "
209             << "): ";
210         cin >> temp;
211         validate(temp, 0, C);
212         fout << temp << " ";
213         cout << "Enter the earliest start (0 <= s <= " << T << "): ";
214         cin >> temp;
215         validate(temp, 0, T);
216         fout << temp << " ";
217         cout << "Enter the latest finish (0 <= f <= " << T << "): ";
218         cin >> temp;
219         validate(temp, 0, T);
220         fout << temp << "\n";
221         cout << "\n\n";
222     }
223 }
```