```cpp
//Fatima Ansari
//BCSF20A046

#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

void validate(unsigned long long int&, int, unsigned long long int);
void storeInputToFile(ofstream&);
int main()
{
    ifstream fin;
    ofstream fout;
    int R, C, F, N, B, T;
    int a, b, x, y, s, f;
    fout.open("inputFile.txt");
    if (fout)
    {
        storeInputToFile(fout);
        fout.close();
    }
    else
    {
        cout << "ERROR!\n";
        cout << "File not Created Successfully!\n";
        exit(0);
    }
    fin.open("inputFile.txt");
    if (!fin)
    {
        cout << "ERROR!!\n";
        cout << "The input File is not found\n";
        exit(0);
    }
    else
    {
        // "\nNow, As we Know that we already stored the valid data to the file\n";
        // "so, no need of validation from getting data back from the file!\n";
        fin >> R >> C >> F >> N >> B >> T;
        // Now to assign rides to the vehicles
        int vehicle = 0;
        int score = 0;
        int rides = 0;
        int simulation = 0;
        fout.open("outputFile.txt");
        if (!fout)
        {
            cout << "Output File not Created Successfully\n";
            exit(0);
        }
        else
        {
            bool finalStep = false; // to check for final step simulation
            while (rides < N && !finalStep)
            {
                int previousRides = rides; // to store the previous rides
                if ((N - previousRides) / (F - vehicle))
                {
                    rides += (N - previousRides) / (F - vehicle);
                    if ((N - previousRides) % (F - vehicle++))
                        rides++;
                }
                else
                {
                    rides += N - previousRides;
                }
                fout << rides - previousRides << " "; // No of rides to each vehicle
                int i = 0, j = 0;                     // current position of each vehicle
                int k = previousRides;
                while (k < rides && !finalStep)
                {
                    simulation = 0;
                    fin >> a >> b >> x >> y >> s >> f;
                    if (i == a && j == b) // check for bonus
                        score += B;       // to increase the total score by bonus B
                    // Now to go to the
                    if (i < a)
                        while (i < a)
                        {
                            i++;
                            simulation++;
                        }
                    else
                        while (i > a)
                        {
                            i--;
                            simulation++;
                        }
                    if (j < b)
                        while (j < b)
                        {
                            j++;
                            simulation++;
                        }
                    else
                        while (j > b)
                        {
                            j--;
                            simulation++;
                        }
                    // Now to handle the earliest start
                    while (simulation < s)
                        simulation++;
                    // Now the vehicle is on the starting position
                    // so, it for destination
                    if (i < x)
                        while (i < x)
                        {
                            i++;
                            simulation++;
                        }
                    else
                        while (i > x)
                        {
                            i--;
                            simulation++;
                        }
                    if (j < y)
                        while (j < y)
                        {
                            j++;
                            simulation++;
                        }
                    else
                        while (j > y)
                        {
                            j--;
                            simulation++;
                        }
                    // now to check for score and latest finish
                    if (simulation < f)
                        score += abs(x - a) + abs(y - b);
                    else if (simulation >= T)
                        finalStep = true;
                    fout << k << " ";
                    k++;
                }
                fout << endl;
            }
            fout.close();
        }
```

```
        cout << "\nTotal Score = " << score << endl
            << endl;
        fin.close();
    }

    return 0;
}
void validate(unsigned long long int& val, int r1, unsigned long long int 
  r2)
{
    while (val < r1 || val > r2)
    {
        cout << "ERROR!! Invalid input\nRe-enter in range of (" << r1 << 
          "-" << r2 << "): ";
        cin >> val;
    }
}
void storeInputToFile(ofstream& fout)
{
    unsigned long long int R, C, F, N, B, T, temp;
    cout << "Enter number of row of the gird (1 <= R <= 10000): ";
    cin >> R;
    validate(R, 1, 10000);
    fout << R << " ";
    cout << "Enter number of columns of the gird (1 <= C <= 10000): ";
    cin >> C;
    validate(C, 1, 10000);
    fout << C << " ";
    cout << "Enter number of vehicles in the fleet (1 <= F <= 1000): ";
    cin >> F;
    validate(F, 1, 1000);
    fout << F << " ";
    cout << "Enter number of rides (1 <= N <= 10000): ";
    cin >> N;
    validate(N, 1, 10000);
    fout << N << " ";
    cout << "Enter Pre-ride bonus for starting the ride on time (1 <= B <= 
      10000): ";
    cin >> B;
    validate(B, 1, 10000);
    fout << B << " ";
    cout << "Enter number of steps in the simulation (1 <= T <= 10^9): ";
    cin >> T;
    validate(T, 1, pow(10, 9));
    fout << T << "\n";
    cout << "The first line of the output file has been successfully 
      stored to the file\n";
    // Now going to the next Step
    // Taking input for each rides
```

```cpp
    cout << "\nNow to take more info about each ride\n\n";
    for (int i = 0; i < N; i++)
    {
        // Getting input for ride i
        cout << "\tRide #" << i << " information\n";
        cout << "Enter row of the start intersection (0 <= a <= " << R << "): ";
        cin >> temp;
        validate(temp, 0, R);
        fout << temp << " ";
        cout << "Enter column of the start intersection (0 <= b <=" << C << "): ";
        cin >> temp;
        validate(temp, 0, C);
        fout << temp << " ";
        cout << "Enter row of the finish intersection (0 <= x <= " << R << "): ";
        cin >> temp;
        validate(temp, 0, R);
        fout << temp << " ";
        cout << "Enter column of the finish intersection (0 <= y <= " << C << "): ";
        cin >> temp;
        validate(temp, 0, C);
        fout << temp << " ";
        cout << "Enter the earliest start (0 <= s < " << T << "): ";
        cin >> temp;
        validate(temp, 0, T);
        fout << temp << " ";
        cout << "Enter the latest finish (0 <= f <= " << T << "): ";
        cin >> temp;
        validate(temp, 0, T);
        fout << temp << "\n";
        cout << "\n\n";
    }
}
```