Assignment tsp using dynamic

```
#include <iostream>
#include <algorithm>
#include <climits>
using namespace std;

const int MAX_CITIES = 10;  // Set a maximum limit for cities
int distanceMatrix[MAX_CITIES][MAX_CITIES];  // Static 2D array for distances
int numCities;  // Number of cities

// Function to calculate the total cost of a specific route
int calculateRouteCost(int route[]) {
   int totalCost = 0;
   for (int i = 0; i < numCities - 1; i++) {
      totalCost += distanceMatrix[route[i]][route[i + 1]];
   }
   totalCost += distanceMatrix[route[numCities - 1]][route[0]];  // Return to starting city
   return totalCost;
}

int main() {
   cout << "Enter the number of cities: ";
   cin >> numCities;

   if (numCities > MAX_CITIES) {
      cout << "Number of cities exceeds the maximum limit of " << MAX_CITIES << ".\n";
      return 1;  // Exit if cities exceed the predefined limit
   }

   // Input the distance matrix
```

```
cout << "Enter the distance matrix:\n";
for (int i = 0; i < numCities; i++) {
    for (int j = 0; j < numCities; j++) {
        cin >> distanceMatrix[i][j];
    }
}

// Create an array to store the cities (0 to numCities-1)
int cities[MAX_CITIES];
for (int i = 0; i < numCities; i++) {
    cities[i] = i;
}

// Find the minimum cost by evaluating all permutations
int minCost = INT_MAX;
int bestRoute[MAX_CITIES];

do {
    int currentCost = calculateRouteCost(cities);
    if (currentCost < minCost) {
        minCost = currentCost;
        copy(cities, cities + numCities, bestRoute);  // Save the best route
    }
} while (next_permutation(cities + 1, cities + numCities));  // Fix city 0 as the starting point

// Output the results
cout << "\nMinimum cost of the Traveling Salesman Problem: " << minCost << endl;
cout << "Path taken: ";
for (int i = 0; i < numCities; i++) {
    cout << bestRoute[i] + 1 << " -> ";
}
```

```
    cout << "1\n";  // Return to the starting city


    return 0;
}
```