DSA Project Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>

#define V 6 // Number of nodes in the graph

// Function prototypes
int minDistance(int dist[], int visited[]);
void dijkstra(int graph[V][V], int src, int dist[]);
void printAsciiPlot(const char* title, double values[]);

// Function to find the vertex with the minimum distance value
int minDistance(int dist[], int visited[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++) {
        if (!visited[v] && dist[v] <= min) {
            min = dist[v];
            min_index = v;
        }
    }
    return min_index;
```

```
}

// Dijkstra's algorithm function
void dijkstra(int graph[V][V], int src, int dist[]) {
    int visited[V] = {0};
    for (int i = 0; i < V; i++) {
        dist[i] = INT_MAX;
        visited[i] = 0;
    }
    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, visited);
        visited[u] = 1;

        for (int v = 0; v < V; v++) {
            if (!visited[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }
}

// Function to calculate horsepower
double calculateHP(int power, int rpm) {
    if (rpm == 0) return 0; // Avoid division by zero
    return (power / (double)rpm) * 5252; // HP = (Power (W) / RPM) * 5252
}

// Function to print ASCII plot
void printAsciiPlot(const char* title, double values[]) {
    printf("\nASCII Plot of %s:\n", title);
    for (int i = 0; i < V; i++) {
        printf("Node %d: ", i);
        int stars = (int)(values[i] / 100); // Scale the horsepower values to a
manageable size
        for (int j = 0; j < stars; j++) {
            printf("*");
        }
        printf(" (%.2f)\n", values[i]);
    }
}

int main() {
    int graph[V][V] = {0}; // Initialize the graph with zeros
    double initialHP[V], tunedHP[V]; // Arrays to hold horsepower values

    // Open the CSV file for initial values
    FILE *initialFile = fopen("D:\\DSA_PROJECT\\Intialvalues.csv", "r");
    if (!initialFile) {
        perror("Failed to open initial_values.csv");
        return 1;
    }
```

```
// Read initial values and calculate horsepower
for (int i = 0; i < V; i++) {
    int power, rpm;
    fscanf(initialFile, "%d,%d", &power, &rpm);
    initialHP[i] = calculateHP(power, rpm);
}
fclose(initialFile);

// Open the CSV file for tuned values
FILE *tunedFile = fopen("D:\\DSA_PROJECT\\FinalValues.csv", "r");
if (!tunedFile) {
    perror("Failed to open tuned_values.csv");
    return 1;
}

// Read tuned values and calculate horsepower
for (int i = 0; i < V; i++) {
    int power, rpm;
    fscanf(tunedFile, "%d,%d", &power, &rpm);
    tunedHP[i] = calculateHP(power, rpm);
}
fclose(tunedFile);

// Print horsepower values for reference
printf("Horsepower Values:\n");
```

```
    printf("Index   Initial HP      Tuned HP\n");
    for (int i = 0; i < V; i++) {
        printf("%d      %.2f    %.2f\n", i, initialHP[i], tunedHP[i]);
    }

    // Calculate differences and assign them to graph edges
    for (int i = 0; i < V; i++) {
        graph[i][i] = 0; // Distance to itself is zero
        for (int j = 0; j < V; j++) {
            if (i != j) {
                graph[i][j] = tunedHP[j] - initialHP[i]; // Difference in horsepower
            }
        }
    }

    // Run Dijkstra's algorithm from source node 0
    int dist[V];
    int source = 0;
    dijkstra(graph, source, dist);

    // Print the ASCII plots
    printAsciiPlot("Initial Horsepower", initialHP);
    printAsciiPlot("Tuned Horsepower", tunedHP);

    return 0;
}
```