Lab 9:

Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Function to perform Selection Sort

void selectionSort(int arr[], int n) {

int i, j, minIdx, temp;

for (i = 0; i < n - 1; i++) {

minIdx = i;

for (j = i + 1; j < n; j++) {

if (arr[j] < arr[minIdx]) {

minIdx = j;

}

}

// Swap the found minimum element with the first element

temp = arr[minIdx];

arr[minIdx] = arr[i];

arr[i] = temp;

}

}

// Function to generate a random array of integers

void generateRandomArray(int arr[], int n) {

for (int i = 0; i < n; i++) {

arr[i] = rand() % 10000; // random integers between 0 and 9999

}

}

// Function to plot the graph using gnuplot

void plotGraph(int n\_values[], double time\_taken[], int num\_tests) {

// Open a pipe to gnuplot

FILE \*gnuplot = popen("gnuplot -persistent", "w");

if (gnuplot == NULL) {

fprintf(stderr, "Error opening gnuplot\n");

return;

}

// Send commands to gnuplot

fprintf(gnuplot, "set title 'Time Complexity of Selection Sort'\n");

fprintf(gnuplot, "set xlabel 'Number of Elements (n)'\n");

fprintf(gnuplot, "set ylabel 'Time Taken (seconds)'\n");

fprintf(gnuplot, "plot '-' with linespoints title 'Time vs n'\n");

// Send the data points to gnuplot

for (int i = 0; i < num\_tests; i++) {

fprintf(gnuplot, "%d %lf\n", n\_values[i], time\_taken[i]);

}

// End the plot

fprintf(gnuplot, "e\n");

fclose(gnuplot);

}

int main() {

// Seed the random number generator

srand(time(NULL));

// Set the values for n (can be modified or read from a file)

int n\_values[] = {5000, 10000, 15000, 20000, 25000}; // example values of n

int num\_tests = sizeof(n\_values) / sizeof(n\_values[0]);

double time\_taken[num\_tests]; // Array to store time taken for each n

// Loop over different values of n

for (int test = 0; test < num\_tests; test++) {

int n = n\_values[test];

// Generate an array of random integers of size n

int \*arr = (int \*)malloc(n \* sizeof(int));

if (arr == NULL) {

printf("Memory allocation failed!\n");

return 1;

}

generateRandomArray(arr, n);

// Record the start time

clock\_t start\_time = clock();

// Call the selectionSort function

selectionSort(arr, n);

// Record the end time

clock\_t end\_time = clock();

// Calculate the time taken (in seconds)

time\_taken[test] = ((double)(end\_time - start\_time)) / CLOCKS\_PER\_SEC;

// Output the results

printf("Time taken to sort an array of size %d: %f seconds\n", n, time\_taken[test]);

// Free the dynamically allocated memory

free(arr);

}

// Plot the graph using gnuplot

plotGraph(n\_values, time\_taken, num\_tests);

return 0;

}