#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

#define MAX 10

#define MAXSTACK 10

int comparisons = 0; // Variabel untuk menyimpan jumlah perbandingan

int swaps = 0; // Variabel untuk menyimpan jumlah pergeseran

struct Stack {

int left;

int right;

};

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

swaps++; // Menghitung jumlah pergeseran

}

// Percobaan 1: Implementasi pengurutan dengan metode Quick Sort Non-Rekursif

void quickSortNonRecursive(int arr[]) {

Stack stack[MAXSTACK];

int top = 0;

int l, r, i, j, x;

stack[top].left = 0;

stack[top].right = MAX - 1;

top++;

while (top > 0) {

top--;

l = stack[top].left;

r = stack[top].right;

while (r > l) {

i = l;

j = r;

x = arr[(l + r) / 2];

while (i <= j) {

while (arr[i] < x) {

i++;

comparisons++; // Menghitung jumlah perbandingan

}

while (arr[j] > x) {

j--;

comparisons++; // Menghitung jumlah perbandingan

}

if (i <= j) {

swap(&arr[i], &arr[j]);

i++;

j--;

}

}

if (l < j) {

top++;

stack[top].left = l;

stack[top].right = j;

}

r = j;

}

}

}

// Percobaan 2: Implementasi pengurutan dengan metode Quick Sort Rekursif

void quickSortRecursive(int arr[], int l, int r) {

int i, j, x;

x = arr[(l + r) / 2];

i = l;

j = r;

while (i <= j) {

while (arr[i] < x) {

i++;

comparisons++; // Menghitung jumlah perbandingan

}

while (arr[j] > x) {

j--;

comparisons++; // Menghitung jumlah perbandingan

}

if (i <= j) {

swap(&arr[i], &arr[j]);

i++;

j--;

}

}

if (l < j)

quickSortRecursive(arr, l, j);

if (i < r)

quickSortRecursive(arr, i, r);

}

// Percobaan 3: Implementasi pengurutan dengan metode Merge Sort

void merge(int arr[], int temp[], int left, int mid, int right) {

int i, j, k;

i = left;

j = mid;

k = left;

while (i <= mid - 1 && j <= right) {

comparisons++; // Menghitung jumlah perbandingan

if (arr[i] <= arr[j])

temp[k++] = arr[i++];

else

temp[k++] = arr[j++];

}

while (i <= mid - 1) {

temp[k++] = arr[i++];

swaps++; // Menghitung jumlah pergeseran

}

while (j <= right) {

temp[k++] = arr[j++];

swaps++; // Menghitung jumlah pergeseran

}

for (i = left; i <= right; i++) {

arr[i] = temp[i];

swaps++; // Menghitung jumlah pergeseran

}

}

void mergeSort(int arr[], int temp[], int left, int right) {

int mid;

if (left < right) {

mid = (left + right) / 2;

mergeSort(arr, temp, left, mid);

mergeSort(arr, temp, mid + 1, right);

merge(arr, temp, left, mid + 1, right);

}

}

int main() {

int arr[MAX];

int temp[MAX];

int choice;

srand(time(NULL));

cout << "Data sebelum terurut: ";

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

arr[i] = rand() % 100;

cout << arr[i] << " ";

}

cout << endl << endl;

cout << "Pilih metode pengurutan:" << endl;

cout << "1. Quick Sort Non-Rekursif" << endl;

cout << "2. Quick Sort Rekursif" << endl;

cout << "3. Merge Sort" << endl;

cout << "Masukkan pilihan Anda (1-3): ";

cin >> choice;

switch (choice) {

case 1:

quickSortNonRecursive(arr);

break;

case 2:

quickSortRecursive(arr, 0, MAX - 1);

break;

case 3:

mergeSort(arr, temp, 0, MAX - 1);

break;

default:

cout << "Pilihan tidak valid!" << endl;

return 0;

}

cout << "Data setelah terurut: ";

for (int i = 0; i < MAX; i++)

cout << arr[i] << " ";

cout << endl;

cout << "Jumlah perbandingan: " << comparisons << endl;

cout << "Jumlah pergeseran: " << swaps << endl;

return 0;

}