Description:

C program that finds the maximum and minimum of an array using a divide-and-conquer approach and reports the CPU time with clock().

Source Code:

```
#include <stdio.h>

#include <time.h>


// Define a structure to hold both maximum and minimum elements

struct pair {

        int max;

        int min;

};

struct pair maxMinDivideConquer(int arr[], int low, int high) {

        struct pair result, left, right;

        int mid;


        // Case 1: Small problem size (only 1 element)

        if (low == high) {

        result.max = arr[low];

        result.min = arr[low];

        return result;

        }


        // Case 2: Small problem size (only 2 elements)

        if (high == low + 1) {
```

```
if (arr[low] < arr[high]) {
result.min = arr[low];
result.max = arr[high];
} else {
result.min = arr[high];
result.max = arr[low];
}
return result;
}

// Case 3: Problem size > 2 (Divide and Conquer)
mid = low + (high - low) / 2; // Calculate middle index

// Divide: Recursively find max and min in the two halves
left = maxMinDivideConquer(arr, low, mid);
right = maxMinDivideConquer(arr, mid + 1, high);

// Conquer: Combine the results

// Find the overall maximum
if (left.max > right.max) {
result.max = left.max;
} else {
result.max = right.max;
}
```

```
    // Find the overall minimum
    if (left.min < right.min) {
    result.min = left.min;
    } else {
    result.min = right.min;
    }

    return result;
}

int main() {
    // Array definition and size calculation
    int arr[] = {6, 4, 26, 14, 33, 64, 46};
    int n = sizeof(arr) / sizeof(arr[0]);

    // Timing variables
    clock_t start, end;
    double cpu_time_used;

    // Start timing
    start = clock();

    // Call the function
    // Pass the full array range: 0 to n-1
    struct pair result = maxMinDivideConquer(arr, 0, n - 1);
```

```
        // Stop timing
        end = clock();

        // Print results
    printf("Maximum element: %d\n", result.max);

    printf("Minimum element: %d\n", result.min);

        // Calculate time
        cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;

        // Print execution time
    printf("Execution time: %.6f seconds\n", cpu_time_used);

        return 0;
}
```

Output:

```
Maximum element: 64
Minimum element: 4
Execution time: 0.000002 seconds
```