LAB-2

2) Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void merge(int B[], int C[], int A[], int p, int q) {
    int i = 0, j = 0, k = 0;
    while (i < p && j < q) {
        if (B[i] <= C[j]) {
            A[k++] = B[i++];
        } else {
            A[k++] = C[j++];
        }
    }
    while (i < p) {
        A[k++] = B[i++];
    }
    while (j < q) {
        A[k++] = C[j++];
    }
}

void mergesort(int A[], int n) {
    if (n < 2) return;

    int mid = n / 2;
    int *B = (int*)malloc(mid * sizeof(int));
    int *C = (int*)malloc((n - mid) * sizeof(int));

    for (int i = 0; i < mid; i++) B[i] = A[i];
    for (int i = mid; i < n; i++) C[i - mid] = A[i];

    mergesort(B, mid);
    mergesort(C, n - mid);
    merge(B, C, A, mid, n - mid);

    free(B);
    free(C);
}

void printArray(int A[], int n) {
```

```c
    for (int i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");
}

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int *A = (int*)malloc(n * sizeof(int));
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    clock_t start,end;
    printf("Original array: ");
    printArray(A, n);

    start=clock();
    mergesort(A,n);
    end=clock();

    printf("Sorted array: ");
    printArray(A, n);
    printf("Time taken = %f", (double)(end-start)/CLK_TCK);
    free(A);
    return 0;
}
```

OUTPUT:

```
PS D:\002> cd "d:\002\" ; if ($?) { gcc 12.c -o 12 } ; if ($?) { .\12 }
Enter the number of elements: 8
Enter the elements: 8 3 2 9 7 1 5 4
Original array: 8 3 2 9 7 1 5 4
Sorted array: 1 2 3 4 5 7 8 9
Time taken = 0.000000
PS D:\002>
```