LAB PROGRAM 6

AIM: Sort a given set of N integer elements using Heap Sort technique and compute its time taken

SOURCE CODE

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>


// Function to swap two elements

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

   int temp = *a;

   *a = *b;

   *b = temp;

}


// Function to heapify a subtree rooted at index i

void heapify(int arr[], int n, int i) {

   int largest = i;

   int left = 2 * i + 1;

   int right = 2 * i + 2;


   if (left < n && arr[left] > arr[largest])

      largest = left;


   if (right < n && arr[right] > arr[largest])

      largest = right;


   if (largest != i) {

      swap(&arr[i], &arr[largest]);
```

```
        heapify(arr, n, largest);
    }
}

// Heap Sort function
void heapSort(int arr[], int n) {
    // Build heap (rearrange array)
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // Extract elements from the heap one by one
    for (int i = n - 1; i > 0; i--) {
        swap(&arr[0], &arr[i]); // Move current root to end
        heapify(arr, i, 0); // Call max heapify on the reduced heap
    }
}

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

    int arr[N];
    printf("Enter %d integers: ", N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    clock_t start_time = clock(); // Start measuring time

    heapSort(arr, N);
```

```
    clock_t end_time = clock(); // End measuring time
    double time_taken = (double)(end_time - start_time) / CLOCKS_PER_SEC;

    printf("Sorted array: ");
    for (int i = 0; i < N; i++) {
        printf("%d ", arr[i]);
    }
    printf("\nTime taken: %f seconds\n", time_taken);

    return 0;
}
```

OUTPUT SCREENSHOT