Experiment 24: Heap Sort

Code:

```
#include <stdio.h>

// Function to heapify a subtree rooted with node i
void heapify(int arr[], int n, int i) {
    int largest = i;      // Initialize largest as root
    int left = 2 * i + 1;  // left child
    int right = 2 * i + 2; // right child

    // If left child is larger than root
    if (left < n && arr[left] > arr[largest])
        largest = left;

    // If right child is larger than largest so far
    if (right < n && arr[right] > arr[largest])
        largest = right;

    // If largest is not root
    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;

        // Recursively heapify the affected subtree
        heapify(arr, n, largest);
```

```
    }
}

// Main function to do heap sort
void heapSort(int arr[], int n) {
    // Build max heap
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // Extract elements from heap one by one
    for (int i = n - 1; i > 0; i--) {
        // Move current root to end
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        // Call heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

// Function to print array
void display(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver code
int main() {
    int arr[100], n;
```

```
    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Original array: ");
    display(arr, n);

    heapSort(arr, n);

    printf("Sorted array (Heap Sort): ");
    display(arr, n);

    return 0;
}
```

Output:

```
Enter number of elements: 6
Enter 6 elements:
60 10 50 20 70 30
Original array: 60 10 50 20 70 30
Sorted array (Heap Sort): 10 20 30 50 60 70


=== Code Execution Successful ===
```