Practical no: 12

Objective:Write a program in C that will sort an array using heap sort.

Program Codes: Following is the code of this problems in C:-

1. Practical12.c

```
#include <stdio.h>

// Function prototypes
void max_heapify(int* arr, int arr_size, int i);

int main() {
    int size;

    // Taking user input
    printf("\nEnter the size of the array: ");
    scanf("%d", &size);
    int arr[size];

    for (int i = 0; i < size; i++) {
        printf("\nEnter Data at arr[%d]: ", i);
        scanf("%d", &arr[i]);
    }

    // Creating max-heap
    for (int i = (size - 1) / 2; i ⩾ 0; i--) {
        max_heapify(arr, size, i);
    }

    // Sorting via pseudo-delete
    for (int i = size - 1; i > 0; i--) {
        // Swap root with the last element
        int temp = arr[i];
        arr[i] = arr[0];
        arr[0] = temp;

        // Re-heapify the reduced heap
        max_heapify(arr, i, 0);
```

```
    }

    // Printing sorted array
    printf("\nSorted array:");
    for (int i = 0; i < size; i++) {
        printf(" %d", arr[i]);
    }

    return 0;
}

// Function to maintain max-heap property
void max_heapify(int* arr, int arr_size, int i) {
    int largest = i;
    int l = 2 * i + 1; // Left child
    int r = 2 * i + 2; // Right child

    // Check if left child exists and is greater than root
    if (l < arr_size && arr[l] > arr[largest]) {
        largest = l;
    }

    // Check if right child exists and is greater than largest so far
    if (r < arr_size && arr[r] > arr[largest]) {
        largest = r;
    }

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

        // Recursively heapify the affected sub-tree
        max_heapify(arr, arr_size, largest);
    }
}
```

Output: Following is the output of the program:-

```
C:\Users\DV yadav\Documents\Programming Files\C,C++ files>gcc Heap_Sort.c && a.exe

Enter the size of the array: 9
Enter Data at arr[0]: 30
Enter Data at arr[1]: 70
Enter Data at arr[2]: 40
Enter Data at arr[3]: 100
Enter Data at arr[4]: 10
Enter Data at arr[5]: 80
Enter Data at arr[6]: 200
Enter Data at arr[7]: 20
Enter Data at arr[8]: 150

Sorted array: 10 20 30 40 70 80 100 150 200
```