1.WRITE A C PROGRAM ON BINARY HEAP.

```
#include <stdio.h>

#include <stdlib.h>

#define MAX_SIZE 100

void maxHeapify(int arr[], int n, int i);

void buildMaxHeap(int arr[], int n);

void insertMaxHeap(int arr[], int *n, int key);

int extractMax(int arr[], int *n);

void printHeap(int arr[], int n);

void minHeapify(int arr[], int n, int i);

void buildMinHeap(int arr[], int n);

void insertMinHeap(int arr[], int *n, int key);

int extractMin(int arr[], int *n);

void maxHeapify(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) {

       int temp = arr[i];

       arr[i] = arr[largest];

       arr[largest] = temp;

       maxHeapify(arr, n, largest);

   }

}

void buildMaxHeap(int arr[], int n) {

   for (int i = n / 2 - 1; i >= 0; i--) {

       maxHeapify(arr, n, i);

   }

}
```

```
void insertMaxHeap(int arr[], int *n, int key) {
    if (*n == MAX_SIZE) {
        printf("Heap is full\n");
        return;
    }
    int i = (*n)++;
    arr[i] = key;
    while (i != 0 && arr[(i - 1) / 2] < arr[i]) {
        int temp = arr[i];
        arr[i] = arr[(i - 1) / 2];
        arr[(i - 1) / 2] = temp;
        i = (i - 1) / 2;
    }
}
int extractMax(int arr[], int *n) {
    if (*n <= 0) {
        printf("Heap is empty\n");
        return -1;
    }
    if (*n == 1) {
        (*n)--;
        return arr[0];
    }
    int root = arr[0];
    arr[0] = arr[--(*n)];
    maxHeapify(arr, *n, 0);
    return root;
}
void minHeapify(int arr[], int n, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && arr[left] < arr[smallest]) {
        smallest = left;
    }
    if (right < n && arr[right] < arr[smallest]) {
        smallest = right;
    }
    if (smallest != i) {
        int temp = arr[i];
        arr[i] = arr[smallest];
        arr[smallest] = temp;
        minHeapify(arr, n, smallest);
    }
}
void buildMinHeap(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        minHeapify(arr, n, i);
    }
}
void insertMinHeap(int arr[], int *n, int key) {
    if (*n == MAX_SIZE) {
        printf("Heap is full\n");
        return;
    }
    int i = (*n)++;
    arr[i] = key;
    while (i != 0 && arr[(i - 1) / 2] > arr[i]) {
        int temp = arr[i];
        arr[i] = arr[(i - 1) / 2];
        arr[(i - 1) / 2] = temp;
        i = (i - 1) / 2;
    }
}
int extractMin(int arr[], int *n) {
    if (*n <= 0) {
        printf("Heap is empty\n");
        return -1;
    }
    if (*n == 1) {
        (*n)--;
        return arr[0];
    }
    int root = arr[0];
    arr[0] = arr[--(*n)];
    minHeapify(arr, *n, 0);
    return root;
}
void printHeap(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int maxHeap[MAX_SIZE] = {10, 20, 15, 30, 40};
    int minHeap[MAX_SIZE] = {40, 30, 20, 10};
    int sizeMax = 5;
    int sizeMin = 4;
    printf("Max-Heap:\n");
    buildMaxHeap(maxHeap, sizeMax);
    printHeap(maxHeap, sizeMax);
    printf("Insert 50 into Max-Heap:\n");
    insertMaxHeap(maxHeap, &sizeMax, 50);
    printHeap(maxHeap, sizeMax);
    printf("Extract Max from Max-Heap:\n");
    printf("Extracted value: %d\n", extractMax(maxHeap, &sizeMax));
    printHeap(maxHeap, sizeMax);
    printf("Min-Heap:\n");
    buildMinHeap(minHeap, sizeMin);
    printHeap(minHeap, sizeMin);
    printf("Insert 5 into Min-Heap:\n");
    insertMinHeap(minHeap, &sizeMin, 5);
    printHeap(minHeap, sizeMin);
    printf("Extract Min from Min-Heap:\n");
    printf("Extracted value: %d\n", extractMin(minHeap, &sizeMin));
    printHeap(minHeap, sizeMin);
    return 0;
}
```

SAMPLE OUTPUT:

```
Max-Heap:
40 30 15 10 20
Insert 50 into Max-Heap:
```

```
50 30 40 10 20 15
Extract Max from Max-Heap:
Extracted value: 50
40 30 15 10 20
Min-Heap:
10 30 20 40
Insert 5 into Min-Heap:
5 10 20 40 30
Extract Min from Min-Heap:
Extracted value: 5
10 30 20 40
```

2.WRITE A C PROGRAM ON HEAP SORTED.

```
#include <stdio.h>
#include <stdlib.h>
void heapify(int arr[], int n, int i);
void buildMaxHeap(int arr[], int n);
void heapSort(int arr[], int n);
void printArray(int arr[], int size);
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 < n && arr[left] > arr[largest]) {
        largest = left;
    }
    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }
    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr, n, largest);
    }
}
void buildMaxHeap(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }
}
void heapSort(int arr[], int n) {
    buildMaxHeap(arr, n);
    for (int i = n - 1; i >= 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array:\n");
    printArray(arr, n);
    heapSort(arr, n);
    printf("Sorted array:\n");
    printArray(arr, n);
    return 0;
}
```

SAMPLE OUTPUT:

```
Original array:
12 11 13 5 6 7
Sorted array:
5 6 7 11 12 13
```