C Program for Binary heap:

```
// C++ program to implement the binary min heap dat
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100
void swap(int* a, int* b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}
void heapify(int* heap, int size, int i)
{
   if (!heap) {
      printf("Invalid Heap!\n");
      return;
   }
   int leftChild = 2 * i + 1;
   int rightChild = 2 * i + 2;
    int smallest = i;
   if (leftChild < size
      && heap[leftChild] < heap[smallest]) {
      smallest = leftChild;
   }
   if (rightChild < size
      && heap[rightChild] < heap[smallest]) {
      smallest = rightChild;
   }
   if (smallest != i) {
      swap(heap + i, heap + smallest);
      heapify(heap, size, smallest);
   }
}
void buildHeap(int* arr, int size)
{
   int start = size / 2 - 1;
   for (int i = start; i >= 0; i--) {
      heapify(arr, size, i);
   }
}
void insert(int* heap, int* size, int element)
{
```

```
    if (*size == MAX_SIZE) {
        printf("Heap Overflow!\n");
        return;
    }
    heap[*size] = element;
    (*size)++;
    int i = *size - 1;
    while (i > 0) {
        if (heap[(i - 1) / 2] > heap[i]) {
            swap(heap + (i - 1) / 2, heap + i);
            i = (i - 1) / 2;
        }
        else {
            break;
        }
    }
}
void delete (int* heap, int* size, int index)
{
    if (size == 0) {
        printf("Heap Underflow\n");
        return;
    }
    heap[index] = heap[*size - 1];
    *size = *size - 1;

    heapify(heap, *size, index);
}
int extractMin(int* heap, int* size)
{
    int min = heap[0];
    delete (heap, size, 0);
    return min;
}
void printHeap(int* heap, int size)
{
    / {
    //        printf("%d ", heap[i + nodeCount])
    //    }
    //    print
    // }
    for (int i = 0; i < size; i++) {
        printf("%d ", heap[i]);
    }
```

```
    printf("\n");
}

// driver code
int main()
{
    int heap[MAX_SIZE] = { 11 };
    int size = 0;

    buildHeap(heap, size);

    insert(heap, &size, 3);
    insert(heap, &size, 2);
    insert(heap, &size, 1);
    insert(heap, &size, 15);
    insert(heap, &size, 5);
    insert(heap, &size, 4);
    insert(heap, &size, 45);

    printHeap(heap, size);

    return 0;
}
```
OUTPUT: 1 3 2 15 5 4 45

C Program for Binary sort:

```
// C Program for HeapSort
#include <stdio.h>

// Heapify function
void heapify(int arr[], int n, int i)
{
        int temp, maximum, left_index, right_index;

        // currently assuming i postion to
        // be holding the largest value
        maximum = i;

        // right child index
        right_index = 2 * i + 2;

        // left child index
```

```
        left_index = 2 * i + 1;

        // if left index value is grater than the current index
        // value
        if (left_index < n && arr[left_index] > arr[maximum])
                maximum = left_index;

        // if right index value is grater than the current index
        // value
        if (right_index < n && arr[right_index] > arr[maximum])
                maximum = right_index;

        // checking if we needed swaping the elements or not
        if (maximum != i) {
                temp = arr[i];
                arr[i] = arr[maximum];
                arr[maximum] = temp;
                heapify(arr, n, maximum);
        }
}

// HeapSorting function
void heapsort(int arr[], int n)
{
        int i, temp;

        // performing heapify on the non leaf nodes so n/2 - 1
        // to 0 are the non leaf nodes
        for (i = n / 2 - 1; i >= 0; i--) {
                heapify(arr, n, i);
        }
        // the current array is changed to max heap

        for (i = n - 1; i > 0; i--) {
                temp = arr[0];
                arr[0] = arr[i];
                arr[i] = temp;
                heapify(arr, i, 0);
        }
}

// Driver code
int main()
{
```

```
	// initializing the array
	int arr[] = { 20, 18, 5, 15, 3, 2 };
	int n = 6;

	// Displaying original array
	printf("Original Array : ");
	for (int i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}

	printf("\n");
	heapsort(arr, n);

	// Displaying sorted array
	printf("Array after performing heap sort: ");
	for (int i = 0; i < n; i++) {
		printf("%d ", arr[i]);
	}
	return 0;
}
```

OUTPUT: Original Array : 20 18 5 15 3 2
Array after performing heap sort: 2 3 5 15 18 20