1.main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "header.h"

int main(){
    heap h;
    init_heap(&h, 10);
    insert_heap(&h, 12);
    insert_heap(&h, 30);
    insert_heap(&h, 44);
    insert_heap(&h, 2);
    insert_heap(&h, 15);
    insert_heap(&h, 90);
    insert_heap(&h, 62);
    insert_heap(&h, 33);
    heap_sort(&h);
    free_heap(&h);

    return 0;
}
```

2.header.h

```
typedef struct heap {
   int *h;
   int size;
   int rear;
} heap;

void init_heap(heap *h, int size);
int parent(int index);
int lchild(int index);
int rchild(int index);
void swap(heap *h, int a, int b);
void print_heap(const heap *h);
void heapify(heap *h, int index);
void heap_sort(heap *h);
void free_heap(heap *h);
void insert_heap(heap *h, int value);
```

3.logic.c

```
#include <stdio.h>
#include <stdlib.h>
#include "header.h"
void init_heap(heap *h, int size){
   h->h = (int*)malloc(size * sizeof(int));
   h->size = size;
   h->rear = -1;
}
int parent(int index) {
   return (index - 1) / 2;
}
int lchild(int index) {
   return 2 * index + 1;
}

int rchild(int index) {
   return 2 * index + 2;
}

void swap(heap *h, int a, int b) {
   int temp = h->h[a];
   h->h[a] = h->h[b];
   h->h[b] = temp;
}
void print_heap(const heap *h) {
   for (int i = 0; i <= h->rear; i++) {
      printf("%d ", h->h[i]);
   }
   printf("\n");
}
```

```
void heapify(heap *h, int index) {
   int largest = index;
   int left = lchild(index);
   int right = rchild(index);
   if (left <= h->rear && h->h[left] > h->h[largest]) {
      largest = left;
   }
   if (right <= h->rear && h->h[right] > h->h[largest]) {
      largest = right;
   }
   if (largest != index) {
      swap(h, index, largest);
      heapify(h, largest);
   }
}
void heap_sort(heap *h) {
   int og_rear = h->rear;
   for (int i = h->rear; i > 0; i--) {
      swap(h, 0, i);
      h->rear--;
      heapify(h, 0);
   }
   h->rear = og_rear;
   print_heap(h);
}
void free_heap(heap *h) {
   free(h->h);
   h->h = NULL;
   h->size = h->rear = 0;
}
```

```
void insert_heap(heap *h, int value) {
    if (h->rear == h->size - 1){
        return;
    }
    h->h[++h->rear] = value;
    int i = h->rear;
    while (i > 0 && h->h[i] > h->h[parent(i)]) {
        swap(h, i, parent(i));
        i = parent(i);
    }
}
```

Output: