```c
#include <stdio.h>
void quickSort(int arr[], int low, int high);
int partition(int arr[], int low, int high);
void swap(int *a, int *b);
void printArray(int arr[], int size);
int main() {
    int arr[] = {34, 7, 23, 32, 5, 62, 32};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Original array:\n");
    printArray(arr, size);
    quickSort(arr, 0, size - 1);
    printf("Sorted array in ascending order:\n");
    printArray(arr, size);
    return 0;
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j < high; j++) {
```

```
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```c
#include <stdio.h>
void quickSortDescending(int arr[], int low, int high);
int partitionDescending(int arr[], int low, int high);
void swap(int *a, int *b);
void printArray(int arr[], int size);
int main() {
    int arr[] = {34, 7, 23, 32, 5, 62, 32};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Original array:\n");
    printArray(arr, size);
    quickSortDescending(arr, 0, size - 1);
    printf("Sorted array in descending order:\n");
    printArray(arr, size);
    return 0;
}
void quickSortDescending(int arr[], int low, int high)
    {
    if (low < high) {
        int pi = partitionDescending(arr, low, high);
        quickSortDescending(arr, low, pi - 1);
        quickSortDescending(arr, pi + 1, high);
    }
}
int partitionDescending(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
```

```
    for (int j = low; j < high; j++) {
        if (arr[j] > pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```c
#include <stdio.h>
typedef struct {
    int *arr;
    int size;
    int capacity;
} MaxHeap;
MaxHeap* createHeap(int capacity);
void insert(MaxHeap *heap, int value);
int extractMax(MaxHeap *heap);
void maxHeapify(MaxHeap *heap, int i);
void swap(int *a, int *b);
void printHeap(MaxHeap *heap);
void freeHeap(MaxHeap *heap);
MaxHeap* createHeap(int capacity) {
    MaxHeap *heap = (MaxHeap *)malloc(sizeof(MaxHeap));
    heap->arr = (int *)malloc(capacity * sizeof(int));
    heap->size = 0;
    heap->capacity = capacity;
    return heap;
}
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void maxHeapify(MaxHeap *heap, int i) {
```

```
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < heap->size && heap->arr[left] > heap
        ->arr[largest]) {
        largest = left;
    }
    if (right < heap->size && heap->arr[right] > heap
        ->arr[largest]) {
        largest = right;
    }
    if (largest != i) {
        swap(&heap->arr[i], &heap->arr[largest]);
        maxHeapify(heap, largest);
    }
}
void insert(MaxHeap *heap, int value) {
    if (heap->size == heap->capacity) {
        printf("Heap is full!\n");
        return;
    }
    heap->size++;
    int i = heap->size - 1;
    heap->arr[i] = value;
    while (i != 0 && heap->arr[(i - 1) / 2] < heap
        ->arr[i]) {
```

```c
int extractMax(MaxHeap *heap) {
    if (heap->size <= 0) {
        printf("Heap is empty!\n");
        return -1;
    }
    if (heap->size == 1) {
        heap->size--;
        return heap->arr[0];
    }
    int root = heap->arr[0];
    heap->arr[0] = heap->arr[heap->size - 1];
    heap->size--;
    maxHeapify(heap, 0);
    return root;
}
void printHeap(MaxHeap *heap) {
    printf("Heap elements:\n");
    for (int i = 0; i < heap->size; i++) {
        printf("%d ", heap->arr[i]);
    }
    printf("\n");
}
void freeHeap(MaxHeap *heap) {
    free(heap->arr);
    free(heap);
}
```

```c
int main() {
    MaxHeap *heap = createHeap(10);
    insert(heap, 10);
    insert(heap, 20);
    insert(heap, 5);
    insert(heap, 30);
    insert(heap, 15);
    printHeap(heap);
    printf("Extracted max: %d\n", extractMax(heap));
    printHeap(heap);
    freeHeap(heap);
    return 0;
}
```