Min Priority Queue

```c
#include <stdio.h>
#include <stdlib.h>

struct MinHeap
{
    int *arr;
    int capacity;
    int size;
};

struct MinHeap *createMinHeap(int capacity)
{
    struct MinHeap *minHeap = (struct MinHeap *)malloc(sizeof(struct
MinHeap));
    minHeap->capacity = capacity;
    minHeap->size = 0;
    minHeap->arr = (int *)malloc(capacity * sizeof(int));
    return minHeap;
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void minHeapifyUp(struct MinHeap *minHeap, int index)
{
    while (index > 0 && minHeap->arr[(index - 1) / 2] > minHeap-
>arr[index])
    {
        swap(&minHeap->arr[(index - 1) / 2], &minHeap->arr[index]);
        index = (index - 1) / 2;
    }
}

void insert(struct MinHeap *minHeap, int data)
{
    if (minHeap->size >= minHeap->capacity)
    {
        printf("Heap is full. Cannot insert.\n");
        return;
    }

    minHeap->arr[minHeap->size] = data;
```

```
    minHeap->size++;
    minHeapifyUp(minHeap, minHeap->size - 1);
}

void minHeapifyDown(struct MinHeap *minHeap, int index)
{
    int leftChild = 2 * index + 1;
    int rightChild = 2 * index + 2;
    int smallest = index;

    if (leftChild < minHeap->size && minHeap->arr[leftChild] < minHeap-
>arr[smallest])
        smallest = leftChild;

    if (rightChild < minHeap->size && minHeap->arr[rightChild] < minHeap-
>arr[smallest])
        smallest = rightChild;

    if (smallest != index)
    {
        swap(&minHeap->arr[index], &minHeap->arr[smallest]);
        minHeapifyDown(minHeap, smallest);
    }
}

int extractMin(struct MinHeap *minHeap)
{
    if (minHeap->size <= 0)
    {
        printf("Heap is empty. Cannot extract minimum.\n");
        return -1;
    }

    int min = minHeap->arr[0];
    minHeap->arr[0] = minHeap->arr[minHeap->size - 1];
    minHeap->size--;
    minHeapifyDown(minHeap, 0);

    return min;
}

void decreaseValue(struct MinHeap *minHeap, int index, int newValue)
{
    if (newValue > minHeap->arr[index])
    {
        printf("New value is greater than the current value. Cannot
decrease.\n");
        return;
```

```
    }

    minHeap->arr[index] = newValue;
    minHeapifyUp(minHeap, index);
}

void display(struct MinHeap *minHeap)
{
    if (minHeap->size == 0)
    {
        printf("Heap is empty.\n");
        return;
    }

    printf("Min-Heap elements:\n");
    for (int i = 0; i < minHeap->size; i++)
    {
        printf("%d ", minHeap->arr[i]);
    }
    printf("\n");
}

void destroyMinHeap(struct MinHeap *minHeap)
{
    free(minHeap->arr);
    free(minHeap);
}

int main()
{
    int capacity;
    printf("Enter the capacity of the Min-Heap: ");
    scanf("%d", &capacity);

    struct MinHeap *minHeap = createMinHeap(capacity);

    int choice, data, index, newValue;

    while (1)
    {
        printf("\nMenu:\n");
        printf("1. Insert\n");
        printf("2. Extract Min\n");
        printf("3. Decrease Value\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        switch (choice)
        {
        case 1:
            if (minHeap->size == minHeap->capacity)
            {
                printf("Heap is full. Cannot insert.\n");
                break;
            }
            printf("Enter data to insert: ");
            scanf("%d", &data);
            insert(minHeap, data);
            break;
        case 2:
            if (minHeap->size == 0)
            {
                printf("Heap is empty. Cannot extract minimum.\n");
                break;
            }
            printf("Extracted Min: %d\n", extractMin(minHeap));
            break;
        case 3:
            if (minHeap->size == 0)
            {
                printf("Heap is empty. Cannot decrease value.\n");
                break;
            }
            printf("Enter index: ");
            scanf("%d", &index);
            if (index < 1 || index >= minHeap->size)
            {
                printf("Invalid index.\n");
                break;
            }
            printf("Enter new value: ");
            scanf("%d", &newValue);
            decreaseValue(minHeap, index - 1, newValue);
            break;
        case 4:
            display(minHeap);
            break;
        case 5:
            destroyMinHeap(minHeap);
            printf("Exiting...\n");
            exit(1);
        default:
            printf("Invalid choice. Please try again.\n");
        }
```

```
    }

    return 0;
}
```