1.main.c

```
#include <stdio.h>

#include <stdlib.h>

#include "header.h"


int main(int argc, char *argv[]) {

    if (argc != 2) {

        printf("Usage: %s <filename>\n", argv[0]);

        return 1;

    }


    const char *filename = argv[1];

    Heap heap;

    initHeap(&heap);


    readFromFile(filename, &heap);

    heapSort(&heap, filename);


    return 0;

}
```

2.header.h

```
// Define the max size for the heap
#define MAX_HEAP_SIZE 100

// Structure to define a Heap
typedef struct {
    int data[MAX_HEAP_SIZE];
    int size;
} Heap;

// Function prototypes
void initHeap(Heap *heap);
void insertHeap(Heap *heap, int value);
int deleteMin(Heap *heap);
void heapify(Heap *heap);
// void heapSort(Heap *heap);
void heapSort(Heap *heap, const char *filename);
void readFromFile(const char *filename, Heap *heap);
```

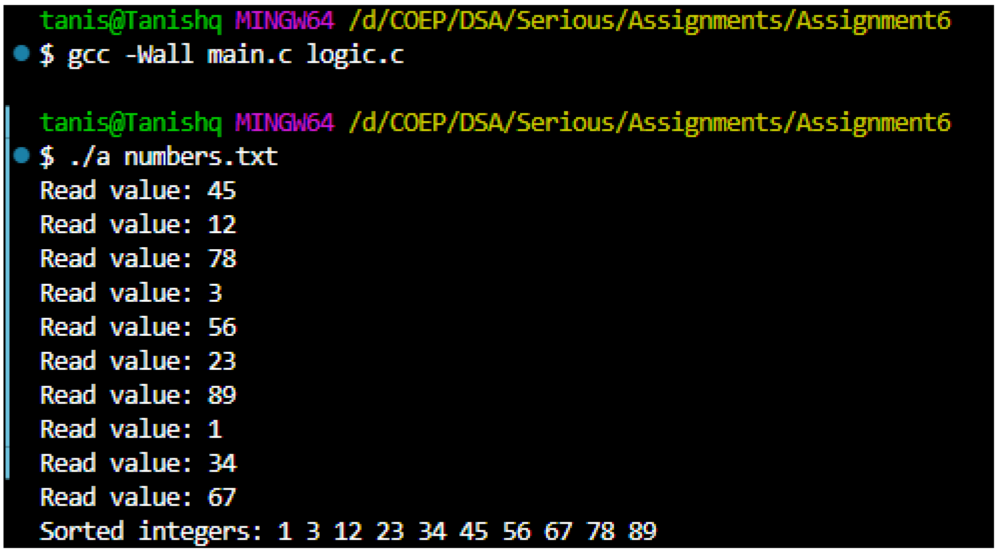
3.logic.c

```
#include <stdio.h>

#include <stdlib.h>

#include "header.h"


// Initialize the heap

void initHeap(Heap *heap) {

   heap->size = 0;

}


// Insert an element into the heap

void insertHeap(Heap *heap, int value) {

   if (heap->size >= MAX_HEAP_SIZE) {

      printf("Heap overflow: Cannot insert %d\n", value);

      return;

   }

   int i = heap->size++;

   while (i > 0 && heap->data[(i - 1) / 2] > value) {

      heap->data[i] = heap->data[(i - 1) / 2];

      i = (i - 1) / 2;

   }

   heap->data[i] = value;

}


// Delete the minimum element from the heap

int deleteMin(Heap *heap) {

   if (heap->size <= 0) {

      printf("Heap underflow\n");

      return -1;

   }

   int min = heap->data[0];
```

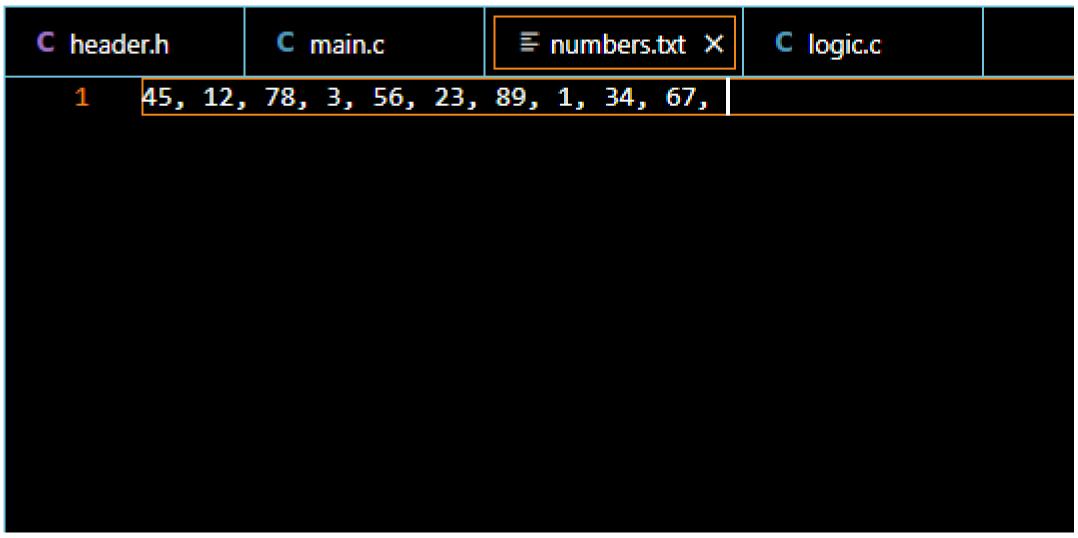
```
    heap->data[0] = heap->data[--heap->size];
    heapify(heap);
    return min;
}


// Maintain the heap property
void heapify(Heap *heap) {
    int i = 0;
    while (i * 2 + 1 < heap->size) {
        int minChild = i * 2 + 1;
        if (minChild + 1 < heap->size && heap->data[minChild + 1] < heap->data[minChild]) {
            minChild++;
        }
        if (heap->data[i] <= heap->data[minChild]) {
            break;
        }
        int temp = heap->data[i];
        heap->data[i] = heap->data[minChild];
        heap->data[minChild] = temp;
        i = minChild;
    }
}
// Sort the heap and display sorted elements
// void heapSort(Heap *heap) {
//     int originalSize = heap->size;
//     printf("Sorted integers: ");
//     for (int i = 0; i < originalSize; i++) {
//         printf("%d ", deleteMin(heap));
//     }
//     printf("\n");
// }
```

```
void heapSort(Heap *heap, const char *filename) {
   int originalSize = heap->size;
   // Open the file in append mode to add sorted and unsorted contents
   FILE *file = fopen(filename, "a");
   if (!file) {
      printf("Error opening file for writing\n");
      return;
   }
   printf("Sorted integers: ");
   fprintf(file, "\nSorted integers: ");
   for (int i = 0; i < originalSize; i++) {
      int minValue = deleteMin(heap);
      printf("%d ", minValue);
      fprintf(file, "%d ", minValue);  // Write each sorted integer to the file
   }
   fclose(file);
}
// Read integers from a file and insert them into the heap
void readFromFile(const char *filename, Heap *heap) {
   FILE *file = fopen(filename, "r");
   if (!file) {
      printf("Error opening file\n");
      exit(1);
   }
   int value;
   while (fscanf(file, "%d,", &value) != EOF) {  // Added comma to handle comma-separated format
      printf("Read value: %d\n", value); // Debugging statement
      insertHeap(heap, value);
   }
   fclose(file);
}
```

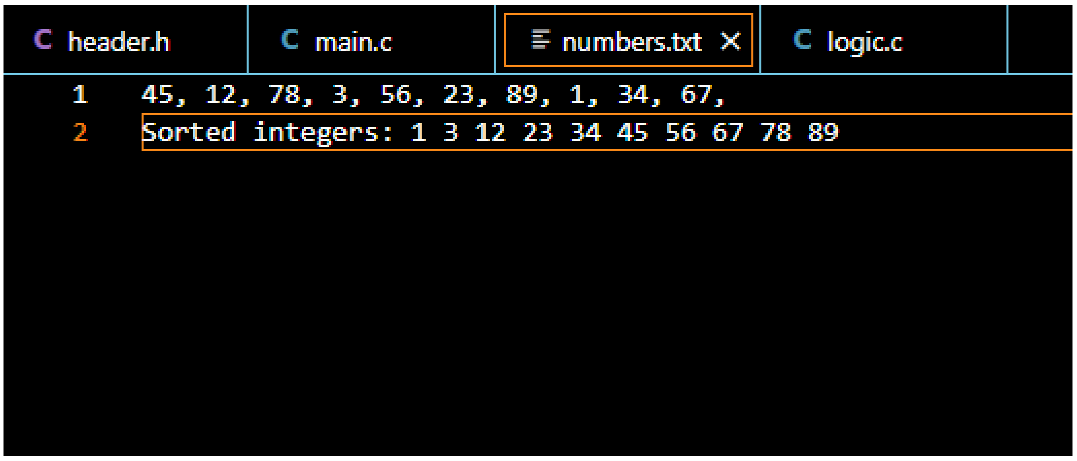
Output:



```
tanis@Tanishq MINGW64 /d/COEP/DSA/Serious/Assignments/Assignment6
$ gcc -Wall main.c logic.c

tanis@Tanishq MINGW64 /d/COEP/DSA/Serious/Assignments/Assignment6
$ ./a numbers.txt
Read value: 45
Read value: 12
Read value: 78
Read value: 3
Read value: 56
Read value: 23
Read value: 89
Read value: 1
Read value: 34
Read value: 67
Sorted integers: 1 3 12 23 34 45 56 67 78 89
```



```
45, 12, 78, 3, 56, 23, 89, 1, 34, 67,
```



```
45, 12, 78, 3, 56, 23, 89, 1, 34, 67,
Sorted integers: 1 3 12 23 34 45 56 67 78 89
```