PLEASE NOTE: heap.h and heap.c from the book were used as is, and hence not included here.

```
#include "heap.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define NUM_PEOPLE 6

typedef struct Person_ {
 const char *name;
 int age;
 double height;
} Person;

/*To populate a Person struct with name,age and height*/
void populatePerson(Person* p,const char* name, int age, double height){
	p->name = name;
	p->age = age;
	p->height = height;
}

/*To compare two Persons such that  smaller name always sit on top of heapSort in which this compare function will be used */
int comparePeopleByDescendingName(const void* key1, const void* key2)
{	Person* pKey1 = (Person*) key1;
	Person* pKey2 = (Person*) key2;
	if(strcmp((char*)pKey1->name,(char*)pKey2->name) > 0)
		return -1;
	else
		return 1;
}

/*To compare two Persons such that lower age remains on top of heapSort in which this compare function will be used */
int comparePeopleByDescendingAge(const void* key1, const void* key2)
{	Person* pKey1 = (Person*) key1;
	Person* pKey2 = (Person*) key2;
	if(pKey1->age > pKey2->age)
		return -1;
	else
		return 1;
}
```

```
/*To compare two Persons such that smaller height remains on top of heapSort in which
this compare function will be used */
int comparePeopleByDescendingHeight(const void* key1, const void* key2)
{	Person* pKey1 = (Person*) key1;
	Person* pKey2 = (Person*) key2;
	if(pKey1->height > pKey2->height)
		return -1;
	else
		return 1;
}


void outputSorted(const Person people[],int numPeople,
 int (* compare)(const void *pKey1, const void *pKey2)){
	Heap heap;
	int i;
	Person* p;
	/*Initialize the heap*/
	heap_init(&heap, compare,free);
	/*Insert the elements from the people array one by one*/
	for(i=0;i<numPeople;i++){
		heap_insert(&heap, people+i);
	}
	i=numPeople-1;
	printf("%s %7s %7s\n","NAME","AGE","HEIGHT" );
	/*Extract the elements from the heap 1-by-1 based on passed compare function*/
	while(i>=0){
		heap_extract(&heap,(void**)&p);
		printf("%5s %5d %5g\n",p->name,p->age,p->height);
		i--;
	}
	/*Destroy the heap*/
	heap_destroy(&heap);

}

int main(){
	Person p[NUM_PEOPLE];
	/*Populate the array "p" with 6 different people*/
	populatePerson(&p[0],"Zara",24,5.5);
	populatePerson(&p[1],"Mira",45,4.9);
	populatePerson(&p[2],"Lara",12,5.3);
	populatePerson(&p[3],"Jeera",40,6.1);
	populatePerson(&p[4],"Yara",25,5.1);
	populatePerson(&p[5],"Bina",65,5.4);
	printf("-----------------------------\n");
	printf("People in Ascending Name\n");
```

```
        printf("------------------------------\n");
        outputSorted(p,NUM_PEOPLE,comparePeopleByDescendingName);

        printf("------------------------------\n");
        printf("People in Ascending Age\n");
        printf("------------------------------\n");
        outputSorted(p,NUM_PEOPLE,comparePeopleByDescendingAge);

        printf("------------------------------\n");
        printf("People in Ascending Height\n");
        printf("------------------------------\n");
        outputSorted(p,NUM_PEOPLE,comparePeopleByDescendingHeight);
        return 0;
}
```

OUTPUT

```
Ram (master *) HeapSortForObjects $ ./hw7
------------------------------
People in Ascending Name
------------------------------
NAME     AGE  HEIGHT
 Bina    65   5.4
Jeera    40   6.1
 Lara    12   5.3
 Mira    45   4.9
 Yara    25   5.1
 Zara    24   5.5
------------------------------
People in Ascending Age
------------------------------
NAME     AGE  HEIGHT
 Lara    12   5.3
 Zara    24   5.5
 Yara    25   5.1
Jeera    40   6.1
 Mira    45   4.9
 Bina    65   5.4
------------------------------
People in Ascending Height
------------------------------
NAME     AGE  HEIGHT
 Mira    45   4.9
 Yara    25   5.1
 Lara    12   5.3
 Bina    65   5.4
 Zara    24   5.5
Jeera    40   6.1
Ram (master *) HeapSortForObjects $
```