CPE 111 – Programming with Data Structures
International Sections, January 2021
Laboratory Exercise 11

Objective
This lab is intended to give you practice writing a program that uses a max-heap to do a heap sort.

Instructions

1. Download all files from the **demos/Lecture11** area of the website. You will find data files **patients1.txt** and **patients2.txt**. These files have the following format:

```
Wichit broken_arm 5
Anchalee burn 4
Somchai bruise 2
Rujirat stab_wound 6
```

Each line has three fields, which are patient name, patient problem and a number that indicates the seriousness of the problem (the problem "severity"), separated by spaces. Bigger severity numbers mean that the patient's problem is more serious or dangerous.

2. Write a program called **emergency.c** which simulates a hospital emergency room. The program will read a file in the above format and store each patient in a max-heap. Then it will extract the patients from the heap one at a time and write their information to an output file. The output file represents the order in which doctors should see each patient. The max-heap will make sure that the patients with the most serious problems get treated first.

3. Your program should take two arguments on the command line, the name of the input file and the name of the output file. For example:

```
./emergency patients1.txt output1.txt
```

4. Your program should *use my implementation of the heap*, **arrayHeap.c**. This means you will need to include **abstractHeap.h** in your program and link **arrayHeap.o** with **emergency.o** to create your executable.

5. Pseudocode for the main function of the program is on the next page. You will need to define a structure to hold information about each patient (and to send to the heap using the **heapInsert** function). Your structure might look something like this:

```
typedef struct
{
   char name[32];       /* first name of the patient *
   char problem[32];    /* reason for coming to the emergency room */
   int severity;        /* indicates how serious the problem is - higher values mean more serious */
} PATIENT_T;
```

6. You will also need to create a comparison function to pass to the **initHeap** function. For example:

```
/* Compare two patients. If patient1 severity is lower than patient2 severity, return -1.
  * If the severity values are the same, return 0.
  * If patient1 severity is higher than patient2 severity, return 1.
int comparePatients(void* patient1, void* patient2)
{
    PATIENT_T* p1 = (PATIENT_T*) patient1;
    PATIENT_T* p2 = (PATIENT_T*) patient2;
    /* Write the rest of the function here .... */
}
```

Use the code in **heapTester.c** as an example to show you how to write the comparison function and how to call **initHeap()**. Remember that you want a max-heap, so you should set the **bMaxHeap** argument to 1.

7. Modify the make file (Makefile) from the Lecture 11 demos to build your program. Test it with both input data files. Upload the following four items: **emergency.c**, **Makefile**, and the two output files associated with **patients1.txt** and **patients2.txt**.

Pseudocode for main function of ***emergency.c***

```
If argc is less than 3
    give an error message - "you must enter both input file and output file"
        exit the program
Endif

Open the input file
If the file pointer is NULL
        give an error message - "can't open input file"
        exit the program
Endif

Initialize the max-heap - check return value to make sure initialization succeeded
If initialization failed
    give an error message - "can't initialize heap"
    exit the program
Endif

While there are lines left in the file (fgets returns something other than NULL)
        Dynamically allocate a PATIENT_T structure (using calloc)
        If allocation failed (calloc returns NULL)
                give error message - "Memory allocation failure"
                exit program
        Endif
        Parse the line read to get name, problem and severity (using sscanf)
        Store the patient data in the structure fields
        Call heapInsert to add the structure to the heap
Endwhile

Close the input file
Open the output file
If the file pointer is NULL
        give an error message - "can't open output file"
        exit the program
Endif

While heapExtract does not return NULL
        Write information from extracted structure into output file (in same format as input file)
        Free the structure
Endwhile

Close the output file

Exit program
```