COE428 Lab 6: Heaps

Heap add, delete and heapSort

1. IMPORTANT: Two week lab

Note that you have two weeks to complete this lab. This lab must be submitted at least 48 hours before the beginning of your next lab period.

2. Prelab preparation

Before coming to the lab you should:

- Read the lab. Try to prepare any questions you may have about the lab.
- Refer to **Lab Guide**.
- Create your lab directory for lab6. (i.e. use **mkdir lab6** within your coe428 directory.)
- Change to your coe428/lab6 and unzip the lab6.zip file with the command:
  unzip /home/courses/coe428/lab6/lab6.zip

- Ensure that you have downloaded properly by entering the command: **`make`**

  No errors should be reported.

Theory: XML tree representation

The XML language is a textual encoding of a tree data structure. The first start-tag matches the last end-tag and represents the root of the tree. Everything between these tags represent the root's children (which may be empty or be trees).

The table below gives some examples.

Description	XML
Tree with root node only	<node> </node>
Tree with root node and 1 child	<node> <node></node> </node>
Tree with root node and 3 children	<node> <node></node> <node></node> <node></node> </node>
Root with 2 children, 2nd child has 1 child	<node> <node></node> <node> <node></node> </node> </node>

Node identification

The XML tree description above does not include information about the nodes. This can be fixed by allowing start-tags to have additional information associated with them.

For example, a tree with a single root node with the information "foo" associated with it can be expressed in XML as <node id="foo"></node>.

Similarly, a tree with a root (value 5) and two children (values 2 and 7) can be expressed in XML as:

```
<node id="5"><node id="2"></node><node id="7"></node></node>
```

(Note: this is also a valid Binary Search Tree.)

Requirements Summary

Your program must:

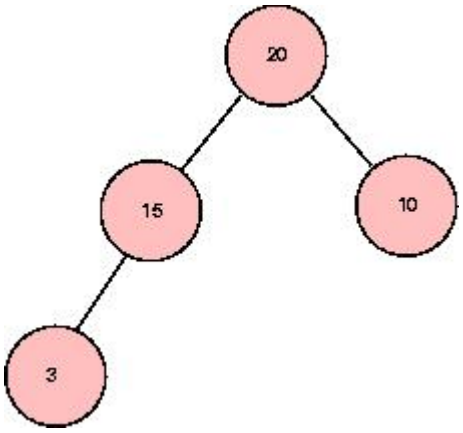
1. Read integers from *stdin* and add each one to a Heap.
2. Print the Heap tree structure as XML.
3. Print the integers in both sorted and reverse-sorted order.

Example

Suppose that the input to the program is:

```
15
20
10
3
```

The tree heap data structure will look like:



The program then prints the tree structure of the heap as an XML expression. The output is:

```
<node id="20"><node id="15"><node id="3"></node></node><node
id="10"></node></node>
```

Next, the program deletes items one-by-one from the heap. As each item is deleted, its value is printed and also pushed onto a stack. The output is the sorted numbers (descending):

```
20

15

10

3
```

Finally, the program pops the stack and prints each item producing the output:

```
3

10

15

20
```

Implementation Notes

To implement the algorithm in C, you need a main() function that reads *stdin* and processes each line. You also need to implement a Heap and a Stack.

You must use separate C source code files for each of these. The files are:

main.c

The main() function in this file reads *stdin* one line at a time. You are provided with a skeleton implementation of main() that handles the "read integers from *stdin* loop".

However, you have to code the actual algorithm. You will need the Heap and the Stack for this.

intStack.c

Again, you are provided with a skeletal version of this file. You need to implement the 3 functions (push, pop and isEmpty). The comments describing what these functions do must not be modified.

Note

The stack you used in the previous lab should work for this lab. Just copy it!

intHeap.c

Again, you are provided with a skeletal version of this file. You need to implement the 3 functions (heapAdd, heapDelete and heapSize). The comments describing what these functions do must not be modified.

Submit your lab

1. Go to your **coe428** directory
2. Zip your **lab6** directory by using the following command:
   zip -r lab6.zip lab6/
3. Submit the lab6.zip file using the following command:

   submit coe428 lab6 lab6.zip

by Ken Clowes, revised by Olivia Das