

Indian Institute of Technology, Guwahati
Department of Computer Science and Engineering
Data Structure Lab (CS210)

Assignment: 8

Date: 20th October, 2016.

Total Marks: 20 (Lab Assignments) + 30 (Offline assignments)

Deadline of Offline Assignment Submission: 26th October, 2016.

Submission Instructions:

- Name of every file should be <Roll_No>_<Question_No>.c. eg. 150101086_1.c, 150101086_2.c and so on.
- Name of the folder you upload should be <Roll_No> eg. 154101086.
- Please do not upload files like a.out, desktop.ini, etc. If there are 'N' questions in assignment, there must be exactly 'N' ".c" or ".cpp" files in the folder. Use only zip to compress the folder to be uploaded.
- You will be awarded ZERO mark if the above rules are violated.
- Please note that you will not be able to submit once the dead line (with 12 hours' late submission) is over.

Lab Assignments:

1. You are required to write two programs, each of which has the same function, but a different underlying algorithm. The programs take as input a file, containing a list of household salaries. The first entry in the file indicates the number of families, followed by the household income of each of these families (each of these entries is separated by a newline). The program reads this file and asks for an input integer k, less than the total number of salaries. The program then outputs the k lowest salaries in ascending order. The three programs implement this functionality in the following three ways:
 - The first program makes a heap of the numbers by inserting the numbers one at a time into the heap and removing the k minimum numbers from the heap (while maintaining the heap). **(10)**
 - The second program builds a heap bottom up and removes the k minimum numbers from the heap (while maintaining the heap). **(10)**

Implement these two algorithms in two different programs. Test these programs on sample input from **test.dat** (provided as separate file).

Offline Assignments:

1. **Implementation of Binomial Heap:** Implementing binomial heaps will require coding the operations: **(30)**
 - **MAKE-BINOMIAL-HEAP**

- **BINOMIAL-HEAP-UNION**
- **BINOMIAL-HEAP-INSERT**
- **BINOMIAL-HEAP-EXTRACT-MIN**

In addition, you need to write the routine showBinomialHeap which displays the binomial heap structure graphically (see below), rotated 90 degrees.

You need to implement a command based interface to test your programs. A sample command is given below

<ul style="list-style-type: none"> • MAKE-BINOMIAL-HEAP • BINOMIAL-HEAP-UNION • BINOMIAL-HEAP-INSERT • BINOMIAL-HEAP-EXTRACT-MIN • Turn off Print after extract min • Turn on print after extract min 	c u i d + -
---	----------------------------

Sample Test Case 1:

```
+-----+
|# create empty heap                                     |
|                                                       |
|# insert 20 keys                                       |
|                                                       |
|# show heap structure                                  |
|Structure of binomial heap (rotated 90 degrees counterclockwise):|
|
|   1     2
|
|       3     4
|
|       5     6
|
|           7     8
|
|       9     10
|
|           11     12
|
|           13     14
|
|               15     16
|
|   17     18
|
|       19     20
|
|# do 10 deletes (extract minimums) with print key flag off
|# show heap structure
```

```

|
|Structure of binomial heap (rotated 90 degrees counterclockwise):|
|
|   13   14
|
|       15   16
|
|       17   18
|
|           19   20
|
|   11   12
|
|# do 10 deletes (extract minimums) with print key flag on
|
|Minimum extracted: 11
|Minimum extracted: 12
|Minimum extracted: 13
|Minimum extracted: 14
|Minimum extracted: 15
|Minimum extracted: 16
|Minimum extracted: 17
|Minimum extracted: 18
|Minimum extracted: 19
|Minimum extracted: 20
|
|# show heap structure (should be empty)
|
|Structure of binomial heap (rotated 90 degrees counterclockwise):|
|
|
|
|# do a delete from empty heap
|Heap Empty
|
|# quit
|
+-----+

```

Sample Test Case 2: See binomialHeapTest.txt

Displaying the Structure of a Binomial Heap

One way to do this is to let `showBinomialHeap` be a non-recursive "driver" that calls a recursive routine `showHeap` to display a binomial heap rotated 90 degrees.

So `showHeap` could be something like:

```

showHeap( x, depth )
    if ( sibling[x] != NIL ) then      |> Do sibling list first
        showHeap( sibling[x], depth )

    |> May have to print a blank line here

```

```

if ( ( child[x] != NIL ) or ( p[x] = NIL ) ) then
    print key[x] shifted 6*depth + 4 spaces      |> First key on a line
    if ( child[x] = NIL ) print a blank line     |> Special case
else
    print key[x] shifted 6 spaces, then a blank line |> Last key on line

if ( child[x] != NIL ) then                    |> Take care of children last
    showHeap( child[x], depth + 1 )

```

And **showBinomialHeap** would be something like:

```

showBinomialHeap(H)
    print "Structure of binomial heap (rotated 90 degrees ccwise):"
    if ( head[H] = NIL ) then print "Empty heap"
                                else showHeap( head[H], 0 )

```

You may have to adjust the formatting to get good looking output.