Design and Analysis of Algorithms 2008 (Home work 3)

October 7, 2008

- Due on Thursday, October 23, before 8 a.m.
- You have a total of 7 late days in the whole term.
- Please give precise arguments for all statements that you write.
- Please do not hesitate to contact me if you do not understand the problems.
- Marks distribution: Problem 1: 10, Prob. 2: 40, Prob 3: 20 and Prob 4: 10.
- Collaboration is encouraged, but you should not copy solutions, but write down your own answers. You should not use any part of code that is not developed by you except the sample codes that I provide. If copying is detected that may immediately lead to a grade less than 7. (**This would be followed strictly**)
- Credits would be given to partial solutions also.
- The answers to questions 1, 3 and 4 should be typed or written clearly and a hard copy is to be submitted. The codes of question 2 should be sent to me by email. Please zip or tar all the files into a single archive before mailing me.

1. Write a non recursive version of the procedure *Heapify*.
2. Implementations (In each of the following the codes should take input from a file and write the output in another file. You should write your codes in C preferably in linux)
   (a) Implement *Heapsort* using recursive version of Heapify. Implement the the procedures parent, left-child and right-child as functions.
   (b) Implement *Heapsort* using the non-recursive version of Heapify as developed in Problem 1. Implement the the procedures parent, left-child and right-child using macros.

(c) Implement *Quicksort* with a randomized pivot.

(d) Implement *Quicksort* using the *qsort* function provided in *stdlib.h*.

3. Time comparisons

   (a) Run the four programs of problem 2 on n randomly generated points where $n = 100, 200, 300, \ldots, 10000$. Measure the running time of all the programs and show their variations with n in a single plot.

   To measure time you may use the following code. The variable timeRequired in the following code will give the number of CPU cycles utilized by your sorting procedure. I shall put a sample code soon in the web-page.

```
 typedef unsigned long long uint64;
 uint64 read_clock_tick()
{ unsigned int h,l;

 __asm__ ( "xor %%eax,%%eax \t \n"
            "cpuid \t \n"
           " rdtsc \t \n"
           "mov %%edx, %0 \t \n"
           "mov %%eax, %1 \t \n"
           :"=m"(h),
           "=m"(l)
           :);
   return((uint64)h<<32) + l;
}

main()
{ uint64 start,end;
  start = read_clock_tick();

  /* Your sorting procedure should be here,
   you should not include any input/output
   operations in this area*/

    end =read_clock_tick();

  timeRequired = end-start;

  printf("The time required for sorting
           = %ld CPU cycles", timeRequired);
}
```

4. Suppose $A = [x_1, x_2, \ldots, x_n]$ is an array of n integers, and let x be an integer. Design an algorithm to find if the sum of any two integers in A is equal to x. Full credit would be given if you can design an algorithm whose running time is $O(n \lg n)$.