

# Cover page for CSC263 Homework #1

(fill and attach this page to your homework)

SUBMITTED BY:

(1) Family Name: \_\_\_\_\_ (2) Family Name: \_\_\_\_\_  
Given Name: \_\_\_\_\_ Given Name: \_\_\_\_\_  
Student Number: \_\_\_\_\_ Student Number: \_\_\_\_\_

Graded Homework should be returned in Tutorial: \_\_\_\_\_(tutorial room number)

FOR HELP WITH YOUR HOMEWORK YOU MAY CONSULT ONLY THE INSTRUCTOR, THE TEACHING ASSISTANTS, YOUR HOMEWORK PARTNER (IF YOU HAVE ONE), YOUR TEXTBOOK AND YOUR CLASS NOTES. **You may not consult any other source.**

*By virtue of submitting this homework I/we acknowledge that I am/we are aware of the policy on homework collaboration for this course.*

Homework Assignment #1  
Due: February 2, 2012, by 5:30 pm  
(in the drop box for this course in Bahen 2220)

1. *On the cover page of your assignment, in addition to your name(s), you must write the location of the tutorial where you want the graded homework to be returned.*
2. *For any question, you may use data structures from class without describing how the operations on these data structures are implemented. You may also use any result (e.g., a worst-case time complexity bound) that we covered in class, or is in the course textbook, by referring to it.*
3. *Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

**Question 1.** (16 marks) Consider the following code for computing the trivial function  $F$  which always returns 0:

```
1. F( integer i )
2.   if odd( i ) then
3.     for k = 0 to i do skip // each ‘skip’ does nothing but takes constant time
4.   return 0
```

Let  $T(n)$  be the worst-case time complexity of executing the above code for  $F$  on an input of size  $n$ . Note that if the input is integer  $i$ , the size of the input is  $n = \lfloor \log_2 i \rfloor + 1$ ; this is because the size of an integer input  $i$  is the number of bits in the binary representation of  $i$ .

Indicate whether  $T(n)$  is  $\Theta(1)$ ,  $\Theta(\log_2 n)$ ,  $\Theta(n)$ ,  $\Theta(n \log_2 n)$ ,  $\Theta(n^2)$ ,  $\Theta(2^n)$ , or  $\Theta(n^n)$ , and prove it. Any answer without a proof will get no credit.

**Question 2.** (14 marks) This question is about MAX-HEAPS.

**a.** (7 marks) You are given the following array  $A$  of 8 integers:  $[5,0,3,1,14,10,16,11]$ .

Apply the linear-time algorithm BUILD-MAX-HEAP(), which is described in the textbook (CLRS 6.3), to the array  $A$  and *show the resulting array*. Do **not** show any intermediate result.

**b.** (7 marks) Starting from an initially *empty* MAX-HEAP array  $A$ , successively insert the keys 5, 0, 3, 1, 14, 10, 16, 11 (one at a time and in that order) into  $A$  using the MAX-HEAP-INSERT() operation which is described in the textbook (CLRS 6.5). *Show the resulting array*. Do **not** show any intermediate result.

**Question 3.** (20 marks) Consider the following additional operations on a MIN-HEAP array  $A$ .

- $\text{CHANGE-KEY}(A, i, key)$ , where  $1 \leq i \leq A.\text{heap-size}$ , changes the priority of element  $A[i]$  to  $key$  and restores the min-heap ordering property.
- $\text{DELETE}(A, i)$ , where  $1 \leq i \leq A.\text{heap-size}$ , deletes the element  $A[i]$  from the heap.

**a.** (14 marks) Give algorithms to implement each of these two operations in worst-case time  $\Theta(\log n)$ , where  $n = A.\text{heap-size}$ . You should describe each algorithm using high-level pseudo-code similar to that used in your textbook, together with a clear explanation in English.

**b.** (6 marks) Explain why the worst-case time complexity of each algorithm is  $\Theta(\log n)$ .

**Question 4.** (20 marks) This question is about the worst-case cost of successively inserting  $k$  elements into a binomial heap (*alias* binomial queue) of size  $n$ .

**a.** (8 marks) Prove that a binomial heap with  $n$  elements has exactly  $n - \alpha(n)$  edges, where  $\alpha(n)$  is the number of 1's in the binary representation of  $n$ .

**b.** (12 marks) Consider the worst-case total cost of successively inserting  $k$  new elements into a binomial heap  $H$  of size  $|H| = n$ . In this question, we measure the worst-case cost of inserting a new element into  $H$  as the maximum number of pairwise comparisons between elements of the binomial heap that is required to do this insertion. In class we proved that for  $k = 1$  (i.e., inserting one element) the worst-case cost is  $O(\log n)$ . Show that if  $k > \log n$ , then the worst-case *total* cost of successively inserting  $k$  elements into  $H$  is only  $O(k)$ . In other words, if  $k > \log n$  then the *average* cost of an insertion (i.e., the worst-case total cost divided by  $k$ ) is bounded by a *constant*.

*Hint:* Note that the cost of each one of the  $k$  consecutive insertions varies — some can be expensive, other are cheaper. Relate the cost of each insertion, i.e., the number of pairwise comparisons that it requires, with the number of extra edges that it forms. Then use part (a).