Computer Science C73 | September 23, 2019
Scarborough Campus | University of Toronto

Homework Assignment #3 — Updated 25/09/19

(worth 4% of the course grade)
Due: September 30, 2019, by 10 am

• *You must submit your assignment as a PDF file through the MarkUs system by logging in with your UTORid at* `https://markus.utsc.utoronto.ca/cscc73f19`. *To work with a partner, you and your partner must form a group on MarkUs.*
• *It is your responsibility to ensure that the PDF file you submit is legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF file you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
• *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
• *For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.*
• *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description.*
• *Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Question 1. (10 marks) Consider the algorithm WEIRDSORT(A, ℓ, r) below, where $A[1..n]$ is an array, and $1 \le \ell \le r \le n$. It turns out that this is a recursive sorting algorithm that (unlike Mergesort) requires no merging and (unlike Quicksort) requires no partitioning.

```
    WEIRDSORT($A, \ell, r$)
1   if $r = \ell$ then                  ▷ $A[\ell..r]$ has length 1
2       return
3   elsif $r = \ell + 1$ then           ▷ $A[\ell..r]$ has length 2
4       if $A[\ell] > A[r]$ then swap $A[\ell]$ and $A[r]$
5       return
6   else                                ▷ $A[\ell..r]$ has length at least 3
7       $t := \lceil 2(r - \ell + 1)/3 \rceil$          ▷ $t$ is two-thirds of the length of $A[\ell..r]$
8       WEIRDSORT($A, \ell, \ell + t - 1$)      ▷ sort the first two-thirds of $A[\ell..r]$
9       WEIRDSORT($A, r - t + 1, r$)      ▷ sort the last two-thirds of $A[\ell..r]$
10      WEIRDSORT($A, \ell, \ell + t - 1$)      ▷ re-sort the first two-thirds of $A[\ell..r]$
11      return
```

a. (5 marks) Prove that a call to WEIRDSORT(A, ℓ, r) will sort the subarray $A[\ell..r]$ in increasing order. To simplify matters, you may assume that the length $r - \ell + 1$ of $A[\ell..r]$ is a power of 3.
b. (4 marks) Use the master theorem to determine the running time $T(n)$ of the call WEIRDSORT$(A, 1, n)$.

c. (1 marks) How does the running time of WEIRDSORT$(A, 1, n)$ compare to the running time of Bubblesort on array $A[1..n]$? Justify your answer.

Question 2. (10 marks) (Based on DPV, Problem 2.25.) In class we discussed Karatsuba's divide-and-conquer algorithm FASTMULT(x, y), which takes as inputs the binary representations x and y of two non-negative integers, and computes the bianry representation of their product.

a. Following is an algorithm (with line 3 left incomplete), which takes as input a positive integer n that is a power of two, and computes the binary representation of the number ten to nth power (i.e., the number that is written in decimal as a 1 followed by n 0s).

```
    POWEROFTENTOBINARY($n$)
1   if $n = 1$ then return "1010" (the binary string that represents the number ten)
2   else
3       $x :=$ ???
4       return FASTMULT($x, x$)
```

Fill in the missing expression in line 3, explain why this algorithm works, and analyse its running time.

b. Following is an algorithm (with line 6 left incomplete), which takes as input the decimal representation x of a non-negative integer whose number of digits is a power of 2, and computes the binary representation of that number. Assume that $x[1..\textbf{length}(x)]$ is a string of digits $0, 1, \ldots, 9$, where $x[1]$ is the most significant and $x[\textbf{length}(x)]$ the least significant digit of the input, and that you have defined an array **binary**$[d]$, which, for any digit d, gives the binary representation of d; i.e., **binary**$[0] = 0$, **binary**$[1] = 1$, **binary**$[2] = 10$, up to **binary**$[9] = 1001$.

```
    DECIMALTOBINARY($x$)
1   if length($x$) = 1 then return binary[$x$]
2   else
3       $n :=$ length($x$)
4       $x_L := x[1..n/2]$
5       $x_R := x[n/2 + 1..n]$
6       return ???
```

Fill in the missing expression in line 6, explain why this algorithm works, and analyse its running time. In constructing the missing expression, you may assume that you are given a function SUM(a, b), which takes as inputs the binary representations a and b of two non-negative integers and returns the binary representation of their sum; and you may also assume that SUM(a, b) runs in time $O\Big(\max\big(\textbf{length}(a), \textbf{length}(b)\big)\Big)$.