Submission Instructions

a. This assignment is to be solved in teams of two students. Take our warning on plagiarism very seriously. *We assume that from the moment of submission you are certifying that it is solely the work of your team, except where explicitly stated otherwise, and that each teammate can individually explain any part of the solution.*

b. Write clear answers. Justify all answers, except where explicitly not required. State any assumptions you make. Thoroughly spell-check and grammar-check your answers. Meticulously follow all instructions in this document and at the homework web-page.

c. Document each algorithm according to the coding convention of the course.

d. Name your report `LastName1.FirstName1 – LastName2.FirstName2.pdf`, but without using any special characters. The report must be in PDF format and must have the cover information of the demo assignment report.

e. Submit your report via the Course Manager server (whose clock may differ from yours) by the deadline given above.

f. *One* teammate must use the group submission feature of the Course Manager, and the other teammate must confirm the submission via the Course Manager before the deadline.

**Failure to follow the instructions above may result in 0 (zero) points**, and we reserve the right to process your solutions automatically.
Question 1: Integer Sort (30 points)

It can be proven that, over arbitrary arrays, the cost in time of any sorting algorithm is $\Omega(n \cdot \lg n)$ in the worst case. In this respect, algorithms like MERGESORT are optimal.

On the other hand, if the arrays to be sorted satisfy certain (additional) properties, then faster algorithms may exist. As an example, consider the following idea for in place sorting of an array $A[1..n]$ of $n$ integers, each one known in advance to belong to the interval $[0..k]$. We will call this algorithm INTEGERSORT.

i. Create an auxiliary integer array $Y[0..k]$. Initialise this array with zeros.

ii. Scan array $A$. At any step, if the current position of $A$ holds value $x$, then increment $Y[x]$ by 1.

iii. To return the ordered array, start with the leftmost position in $A$. Scan $Y$ from left to right. At any position $x$, if $Y[x] = t$ where $t > 0$, then record value $x$ into array $A$ a total of $t$ times, moving rightwards through $A$.

Given the high-level description above:

a. Write pseudocode for INTEGERSORT, having as arguments an array $A[1..n]$ and an integer $k$. The elements in array $A$ are assumed to belong to the integer interval $[0..k]$.

b. Compute the worst-case running time of the algorithm in terms of $\Theta$ notation. Note that the size of the input is made of two numbers, $n$ and $k$.

c. What is the worst-case running time of INTEGERSORT if $k = O(n)$?

Question 2: Binary Multiplication (35 points)

Consider the problem of computing the product of two integers, represented in binary notation as two arrays $A[1..n]$ and $B[1..n]$ of $n$ digits in $\{0,1\}$, with $A[n]$ and $B[n]$ representing the least significant digits.

Input: Two 0/1 arrays $A[1..n]$ and $B[1..n]$ of the same length $n$.

Output: A 0/1 array $C[1..2 \cdot n]$ encoding the number in binary notation, resulting from multiplying the integers encoded in $A$ and $B$.

a. Write pseudocode for a recursive version of the problem, following the divide-and-conquer approach. (Hint: split the two arrays into two halves, recursively multiply the halves in a proper way, and then proceed by merging the partial results using summations and shifts.)

b. Make an educated guess on the worst-case running time of the algorithm in terms of $\Theta$ notation, using the recursion-tree method.

c. Prove your guess by using the substitution method. If your conjecture is not true, then return to step b.

d. Briefly discuss whether the Master Theorem (MT) is applicable to your algorithm or not. In the positive case, check that the answer obtained by applying MT is the same you obtained in steps b and c.
Question 3: The Birthday Present Problem (35 points)

Given a set $S$ of prices of items in a gift shop, as well as an amount $t$ of money collected by a group of people toward buying birthday presents for a common friend, the objective is to determine whether there exists a subset $S' \subseteq S$ whose sum is exactly $t$ (since we do not want to distribute any excess money back to the donors). For example, if $S = \{1, 2, 7, 32, 56, 234, 12332\}$ and $t = 299$, then the subset $S' = \{2, 7, 56, 234\}$ is a solution, but there is no solution for $t = 11$. Perform the following tasks:

a. Justify why dynamic programming is applicable to the problem of determining the existence of $S'$.

b. Design a dynamic programming algorithm to determine whether there exists a subset $S'$.

c. Modify the algorithm from step b to return a subset $S'$, if it exists.

d. Compute an asymptotic upper bound to the running time of your algorithm.