Faculty of Science

Course:	CSCI 3070U: Analysis and Design of Algorithms
Assignment:	#2
Topic:	Sorting, Dynamic Programming, Greedy Algorithms

Note: This assignment is to be completed either individually or in a team of two students. Groups larger than two will not be permitted, because it is almost a guarantee that some student in the team is not meeting all the learning objectives.

Part 1 (10 points)

Suppose you have a log of length *L*, marked to be cut in *n* different locations labeled *1, 2, . . . , n*. The woodcutter will cut a given log of wood, at any place you choose, for **a price equal to the length of the given log**. For simplicity, let indices 0 and n + 1 denote the left and right endpoints of the original log of length *L*. Let d_i denote the distance of mark *i* from the left end of the log and assume that $0=d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1} = L$. Determining the **sequence** of cuts to the log that will cut the log at all the *n* marked places and minimize your total payment. Choose the following strategies:

A. Greedy algorithm picking the point closest to the center of log (**3 points**). This strategy is not optimum.
B. Dynamic programming (**7 points**). The algorithm time-complexity is $O(n^2)$.

Use C, C++, Java, or Python to implement Part A and B (ask the instructor if you have another programming language in mind).

Part 2 (5 points)

For this part of the assignment, you will implement a radix sort procedure for sorting numbers between 0 (inclusive) and 1,000,000 (exclusive) (i.e., 6-digit numbers).

You may find the following code (which determines the value of an arbitrary digit) useful:

```
function getDigit(num, digit) {
        working = num / 10^(digit-1);
        return working mod 10;
}
```

Part 3 (5 points)

Write an implementation of Huffman coding, which is a greedy implementation of assigning prefix codes. This will require a program that does the following:

1. Read through a simple ASCII text file (passed as an argument), determining the frequency of each character in the file
2. Use these frequencies to calculate the optimal prefix codes for each character (Huffman)
    - Print a complete table of prefix codes for the characters in the document (print only characters with frequencies > 0)
3. Normally, you would then encode each character using its prefix code
    - However, actually implementing this requires some trick bit manipulation (which, for some languages, is difficult to do)
    - Instead, calculate the length of the entire document before and after using the prefix codes

Use C, C++, Java, or Python to implement the solution (ask the instructor if you have another programming language in mind).

How to Submit

Put your source code answers to part 1-3 into a ZIP file, called Assignment2_FirstNameLastName_StudentNum.zip (do not use .rar, .7z or other archival formats) and submit this file to Canvas.