Introduction to Computing: Tutorial 1

BSDS I Year, 2026–2027

September 3, 2026

1. You are given the frequencies of occurrence of n data points. Write a Python program to draw a bar diagram / histogram for this data on the terminal. Each bar should be constructed using the `#` character, with the number of such characters representing the height of the bar.
   (a) The bars should first be drawn horizontally.
   (b) Modify your program so that the bars are drawn vertically.

   Example: If your data points are a_0, a_1, a_2, a_3 and their respective frequencies are 2, 3, 1 and 4, your bar diagrams should look like the following.

```
##                                  #
###                          #       #
#                      #     #       #
####                   #     #   #   #
```

   In the diagram on the left, the bars are drawn horizontally, and correspond to a_0, a_1, a_2, a_3, resp., from top to bottom. Thus, the lengths of the bars are 2, 3, 1 and 4, resp. In the diagram on the right, the bars are drawn vertically and are separated by blanks to make the diagram a little easier to understand. The heights of the bars are 2, 3, 1 and 4.

2. *Pattern drawing:* write Python programs to draw the following patterns on the screen, using the `*` character.
   (a) A diamond, given its maximum width n. You may assume that n is an odd number.
   (b) An isosceles trapezium, given l_t and l_b, its top and bottom width. You may assume that $|l_t - l_b|$ is even, and that the height of the trapezium will be $|l_t - l_b|/2$.
   Generalise your program so that it will work for any specified height h, as long as $|l_t - l_b|/h$ is an even integer.

3. Write a Python program that implements the following algorithm to find the square root of any given non-negative integer. Your answer should be correct up to 4 decimal places.

   Algorithm:

   I. Start with a guess, g.
   II. If $g \times g$ is close enough to x, stop and report g as the answer.
   III. Otherwise, generate a new guess by averaging g and x/g.
   IV. Repeat the process until your guess is close enough.

Additional exercises

(a) Mathematically verify whether this procedure will work correctly for *all* non-negative numbers (not just integers).

(b) Modify your program so that the user can specify n, such that the answer is correct up to n decimal places.

4. Write a program to compute the lowest common multiple (LCM) of a given set of positive integers.

   Input format: An integer $n \geq 2$, followed by n more positive intgers, x_1, x_2, ..., x_n.

   Output format: The LCM of x_1, x_2, ..., x_n.

   Sample input 1: `4 2 3 4 5`

   Sample output 1: `60`

5. Let N be a k-digit non-negative integer, and let s_1 be the sum of the k digits of N. If s_1 consists of more than 1 digit, we denote the sum of the digits of s_1 as s_2. In general, for each s_i ($i \geq 1$), we let s_{i+1} denote the sum of the digits of s_i. Note that the sequence $\{s_1, s_2, s_3, \ldots\}$ converges. The limit of this sequence is called the *digital root* of N.

   Write a program that takes a non-negative integer N as input, and computes its digital root.

   Input format: A single non-negative integer N.

   Output format: A single digit, corresponding to the digital root of N.

   Sample input 1: `9875`

   Sample output 1: `2`

   Explanation: $9 + 8 + 7 + 5 = 29$; $2 + 9 = 11$; $1 + 1 = 2$.

   Additional exercise: We have not covered strings in detail yet, but think about how you would modify your program to handle integers that are too large to store using Python's built-in integer type.