Scores

- Each problem is scored either 0, 1, or 2 for insufficient, reasonable, or excellent
- In order to receive a score of 1, the program must meet the following criteria:
  - It must be free of all syntax errors, and of all run-time errors due to uninitialized variables, bad function calls, etc.
  - It must obey at least half of the style guidelines in the general section of the project grading guide
  - It must meet any additional problem-specific criteria for “reasonable” outlined below
- In order to receive a score of 2, the program must meet the criteria above plus the following criteria:
  - It must produce correct output every time it is run
  - It must obey all or nearly all style guidelines in the general section of the project grading guide
  - It must meet additional problem-specific criteria for “excellent” outlined below, if any
- All programs that do not fall into the above categories will receive a score of 0

1. **Cube Roots**

**Additional “Reasonable” Requirements**

- Initial guess for $L$ is always within the closed interval $[1.2]$
- A `while` loop is used, with the terminating condition which will become true when two successive approximation are within a tolerance of $10^{-12}$
- A formula for refining $L$ is used, and this formula provides for overall improvement in estimates of the cube root as the loop iterates
- At least one additional variable is used to store the previous value of $L$, so that the previous and current value can be compared at each iteration
- The cube root estimate and relative error are printed during each iteration of the loop

**Additional “Excellent” Requirements**

- Initial guess for $L$ uses an interpolation function $f$ such that $f(1) = 1$, and $f(8) = 2$
- The approximation converges in under 200 steps

2. **More Flexible DNA Search with Run Information**

**Additional “Reasonable” Requirements**

- An attempt is made to compute `start` as an $L$-digit number, with each digit corresponding to one of four random nucleotides
- A `while` loop is set up to terminate when the current `slice` matches the `start` sequence, or when 10000 characters have been printed
- With the reading of each nucleotide, each `slice` is updated according to the formula resembling

  \[ \text{slice} = 10 \times \text{mod}\left(\text{slice}, 10^{\left(L-1\right)}\right) + \text{next}; \]

  where `next` is the integer corresponding to the next nucleotide
- A variable is used to keep track of the last seen nucleotide, and an attempt is made to update the current run length when the `next` nucleotide is the same as the one last seen, or to reset the run length to zero when they are not the same
- The length of the longest run and the total sequence length are output in the end of the program