Q1. The puzzle called The Tower of Hanoi consists of three pegs, one of which contains several rings stacked in order of descending diameter from bottom to top. The problem is to move the stack of rings to another peg. You are allowed to move only one ring at a time, and at no time is a ring to be placed on top of a smaller one. Observe that if the puzzle involve only one ring, it would be extremely easy. Moreover, when faced with the problem of moving several rings, if you could move all but the largest ring to another peg, the largest ring could then be placed on the third peg, and then the problem would be to move the remaining rings on top of it. Using this observation, develop a recursive algorithm for solving the Tower of Hanoi puzzle for an arbitrary number of rings.

Q2. Design an algorithm to find the square root of a positive number by starting with the number itself as the first guess and repeatedly producing a new guess from the previous one by averaging the previous guess with the result of dividing the original number by the previous guess. Analyze the control of this repeated process. In particular, what condition should terminate the repetition?

Q3. The factorial of 0 is defined to be 1. The factorial of a positive integer is defined to be the product of that integer times the factorial of the next smallest nonnegative integer. We use the notion $n!$ to express the factorial of the integer $n$. Therefore, the factorial of 3 (written 3!) is $3 \times (2!) = 3 \times (2 \times (1!)) = 3 \times (2 \times (1 \times (0!))) = 3 \times (2 \times (1 \times (1))) = 6$. Design a recursive algorithm that computes the factorial of a give value.

Q4. Design an algorithm that, given a list of names, finds the longest name in the list. Determine what your solution does if there are several “longest” names in the list. In particular, what would your algorithm do if all of the names had that same length?

Q5. Using big-theta notation, classify the traditional grade school algorithms for addition and multiplication. That is, if asked to add two numbers each having $n$ digits, how many individual additions must be performed? If requested to multiply two $n$-digit numbers, how many individual multiplications are required?

Q6. Design a procedure for concatenating two linked lists.

Q7. The table below represents a stack stored in a contiguous block of memory cells, as discussed in the text. If the base of the stack is at address 10 and the stack pointer contains the value 12, what value is retrieved by a pop instruction? What value is then in the stack pointer?

| Address | Contents |
|---|---|
| 10 | F |
| 11 | C |
| 12 | A |
| 13 | B |
| 14 | E |

Q8. The table below represents a tree sorted in a machine's memory. Each node of the tree consists of three cells. The first cell contains the data (a letter), the second contains a pointer to the node's left child, and the third contains a pointe to the node's right child. A value of 0 represents a NIL pointer. If the value of the root pointer is 55, draw a picture of the tree.

| Address | Contents |
|---|---|
| 40 | G |
| 41 | 0 |
| 42 | 0 |
| 43 | X |
| 44 | 0 |
| 45 | 0 |
| 46 | J |
| 47 | 49 |
| 48 | 0 |
| 49 | M |
| 50 | 0 |
| 51 | 0 |
| 52 | F |
| 53 | 43 |
| 54 | 40 |
| 55 | W |
| 56 | 46 |
| 57 | 52 |

Q9. Describe a tree structure that can be used to store the genealogical history of a family. What operations are performed on the tree? If the tree is implemented as a linked structure, what pointers should be associated with each node? Design procedures to perform the operations you identified above, assuming that the tree is implemented as a linked structure with the pointers you just described.

Q10. Draw picture showing how the array below appears in a machine's memory when stored in row major order and in column major order:

| | | | |
|---|---|---|---|
| A | B | C | D |
| E | F | G | H |
| I | J | K | L |