Programming Paradigms Lab Assignment (CS453)

Assignment 1 : Recursion

Time : Two weeks

Write Recursive Functions in C for each of the tasks stated below. Also write main functions to demonstrate each of those functions.

Background Theory
Definition: An algorithm or function is recursive if it refers (calls) itself directly or indirectly. Eg,

1. function F1 calls F1 directly
2. function F1 calls F2 and F2, in turn, calls F1. F1, thus, calls F1 indirectly.

Recursion is a totally different method of solving problems. Conventional problem solving methods consists of decomposing the solution into steps, and executing each step in turn. The recursive way of solving a problem, on the other hand, is to reduce the problem into another problem that is exactly of the same type but simpler, and solving the new one instead. And how to solve the new one? It is just reduced into another simpler problem of exactly the same type, and so on, and so on, and so on...

Principle behind designing (developing) recursive algorithm (function):

Step 1. If the problem is too simple, solve it straight way [Basis Step]

Step 2. Otherwise beak the problem into similar but simpler problem with the target that at some point of time it will be simple enough to be solved straight way. [Recursion Step]

Problems

1. Factorial:Factorial of an integer n

If n = 1 then the result is 1 [Step 1]

Else the result is (n X Factorial of (n-1)) [Step 2]

2. Sum of n integers:

If n = 1 then the sum is the given integer itself [Step 1]

Else the sum is (1st integer + sum of remaining (n-1) integers) [Step 2]

3. String comparison:Compare strings s1 and s2 - print 0 if both are same, print -1 if alphabetically s1 comes before s2 and print 1 if alphabetically s1 comes after s2.

If s1 and s2 both are empty strings then print 0. If the 1st character of s1 alphabetically comes before the 1st character of s2 print -1. If the 1st character of s1 alphabetically comes after the 1st character of s2 print 1. [Step 1]

Else compare the rest of s1 with rest of s2 (that is, other than the 1 st characters, the remaining parts of s1 and s2,) [Step 2]

4. Find largest among n integers

If n = 1 then the single integer itself is the largest [Step 1]

Else the largest integer is the larger one between the last integer and largest among the remaining integers. [Step 2]

5. Find the HCF of two integers n1 and n2 (n1 >= n2)

If n1 % n2 = 0, that is n1 divides n2 then n1 is the HCF [Step 1]

Else the desired HCF is the HCF of two integers n2 and (n1 % n2) [Step 2]

6. Search for a given integer num in a given list L of n integers (returns true if num occurs in L, otherwise returns false).

If n = 0, that is L is empty, then return false. If the first item of L is num then return true. [Step 1] Else search for num in the remaining list of (n-1) integers (that is L without the first item) [Step 2]

7. Print a given list L of n items in reverse order

If n = 0, that is, L is empty, then do not print anything. [Step 1]

Else print the last item followed by print the remaining list of (n-1) items in reverse order (that is L without the last item) [Step 2]

8. Find out whether the given string S of length n is a palindrome or not (returns true if S is a palindrome, returns false otherwise). [A palindrome is a string if it reads same from both the ends. Eg. ABCDCBA]

If n=0 or n=1, that is, S is empty or has a single character, then return true. [Step 1]

If the first and last characters of S are different then return false, else find out whether the remaining string (S without the first and last characters) is a palindrome or not. [Step 2]

9. Replace all occurrences of the character c1 in the string S by the character

c2. If S is empty, then, do not do anything. [Step 1]

Else

if the first character of S is c1, then replace it by c2. Replace all occurrences of the character c1 in the remaining string (S without the first character) by the character c2. [Step 2]

10. Sorting an array A of n integers in ascending

order. If n is 1, then, do not do anything. [Step 1]

Else

Find i so that A[i] is largest and then interchange A[i] and A[n] and Sort first n-1 elements of A.
[Step 2]

[Should be attempted once 1 to 10 are done]

11. Let start point to the first node of a singly linked list. SortLL(start) returns a pointer to the first node of the sorted linked list. [Each node contains a integer key on which sorting has to be performed.]

12. Let T be a tree of degree n stored in an array A. Let the nodes of T store positive integers. findLargestNode(A, n) returns the largest integer stored in T.