Lab3- Recursion

You can select and do some questions according to your ability. We would like to note you that the more questions you do the better for you in doing final practical and writing exams.

Write Java programs to perfom the following tasks:

1) Write a recursive function that computes the sum of all numbers from 1 to n, where n is given as parameter.
   //return the sum 1+ 2+ 3+ ...+ n
   int sum(int n)
2) Write a recursive function that finds and returns the minimum element in an array, where the array and its size are given as parameters.
   //return the minimum element in a[]
   int findmin(int a[], int n)
3) Write a recursive function that computes and returns the sum of all elements in an array, where the array and its size are given as parameters.
   //return the sum of all elements in a[]
   int findsum(int a[], int n)
4) Write a recursive function that determines whether an array is a palindrome, where the array and its size are given as parameters.
   //returns 1 if a[] is a palindrome, 0 otherwise. The string a is palindrome if it is the same as its reverse.
   int ispalindrome(char a[], int n)
5) Write a recursive function that searches for a target in a sorted array using binay search, where the array, its size and the target are given as parameters.
6) Implement the Greatest Common Divisor algorithm as recursive method **GCD**. Use recursion. Do NOT use a loop.

$$GCD(m,n)\begin{cases} m & if\ n == 0 \\ GCD(n,\ m\%n) & if\ n > 0 \end{cases}$$

7) Implement the ***power*** function recursively

$$\text{power}(x,\ n)\begin{cases} 1 \text{ if } n = 0 \\ x \cdot power\ (x,\ n-1) \text{ if } x \geq 1 \end{cases}$$

8) Implement the factorial function recursively as ***fact***

$$\text{fact}(n)\begin{cases} 1 \text{ if } n \leq 1 \\ x \cdot fact\ (n-1) \text{ if } x > 1 \end{cases}$$

9) Implement Fibonacci recursively as ***fib***

$$\text{fib}(n) \begin{cases} 1 \text{ if } n \leq 2 \\ \text{fib}(n-1) + \text{fib}(n-2) \text{ if } x \geq 1 \end{cases}$$

10) Write recursive method ***addReciprocals*** that takes an integer as a parameter and returns the sum of the first n reciprocals.
addReciprocals(n) returns (1.0 + 1.0/2.0 + 1.0/3.0 + 1.0/4.0 + ... + 1.0/n).

11) Tree height. Given a labeled binary tree (represented by a pointer to a TreeNode) calculate its height.

12) Tree size. Given a labeled binary tree (represented by a pointer to a TreeNode) calculate its size.