# t09 Recursion

1. The recursive function fib()computing Fibonacci number is very inefficient as it computes the same values many times.

```
int fib (int n) {
        if (n <= 2)   return 1;
        else   return fib ( n-1 ) + fib ( n-2 );
}
```

This inefficiency can be rid of by using the technique called memorization. Assuming n < 50 and we use an array f of 50 entries initialized with 0. Whenever fib(k) is computed, it is stored in f[k]. When fib(m) is called, if f[m] is found to be non-zero, return f[m] immediately. Rewrite the recursive fib() to make use of memoization. You may need to implement a helper function fibInit() to perform the proper initialization of the array f before calling the modified fib(). Implement a driver as well. What is the largest value of n can be for fib(n) to work?

2. Write some functions to check recursively whether a given string is an Id. An Id is a letter ('a' to 'z', 'A' to 'Z', and '_'), or an Id followed by a letter, or an Id followed by a digit. Note that an empty string is not an Id.

3. Implement the method **int** reverse(…) in C++ to return the reverse of a given integer (int). For example, the reverse of 123 is 321. The reverse of 3210 is 123. The method must be recursive, though you are free to specify the list of parameters of this method. What is the complexity of the method reverse?

4. The recursive function recVerify will verify that a given int array is sorted into ascending order.
(a) Write recVerify and its test driver.
(b) What is its time complexity?