18 The runtime stack, recursion explained and simulated

Here is a recursive version of `gcd()`.

```
int gcd(int m, int n) // assume m>=n>=0 and m>0; not all necessary
{
  if (n==0)
    return m;
  else
  {   int p; // not essential, makes things clearer
    p = gcd ( n, m-n );
    return p;
  }
}
```

```
gcd(40,25) will call
gcd(25,15) which calls
gcd(15,10) which calls
gcd(10,5)  which calls
gcd(5,0)
gcd(5,0) returns 5 to the calling routine
gcd(10,5) returns 5 to the calling routine
gcd(15,10) returns 5 to the calling routine
gcd(25,15) returns 5 to the calling routine
gcd(40,25) which returns 5 to its calling routine.
```

In general, when routine A calls routine B, then when B terminates, A is resumed. There is a *nesting effect*. To show the nesting pattern, we shall use indentation.

```
gcd(40,25) calls
  gcd(25,15): gcd(40,25) pauses and gcd(25,15) calls
    gcd(15,10) gcd(40,25), gcd(25,15) both paused, next call
      gcd(10,5) gcd(40,25), gcd(25,15), gcd (10,5) paused, next call
        gcd(5,0)
        gcd(5,0) returns and
      gcd(10,5) resumes, returning 5
    gcd(15,10) resumes returning 5
  gcd(25,15) resumes, returning 5
gcd(40,25) resumes, returning 5
```

So, at any time a particular routine is operating, and some others are paused, incomplete. Again:

```
int gcd(int m, int n) // assume m>=n>=0 and m>0; not all necessary
{
```

```
  if (n==0)
    return m;
  else
  {   int p; // not essential, makes things clearer
    p = gcd ( n, m-n );
    return p;
  }
}
```

There are two arguments and one local variable, that is, when `gcd()` is called there are three variables of interest:

```
    m       n      p
```

The runtime stack. There is a section of memory called the *runtime stack* in which the local variables of the routine operating and any incomplete, paused, routines, are stored. The variables connected with a particular routine are stored in a *stack frame*. When the routine begins, the frame is created and initialised. When it ends, the frame is removed, exposing the frame of the calling routine.

We illustrate them with indenting. This time the calls are on the right.

```
  m  n  p   m  n  p  m  n  p   m  n  p   m  n  p
|40,25,--|.....................................gcd(40,25)
          |25,15,--|.............................  gcd(25,15)
                   |15,10,--|.....................    gcd(15,10)
                            |10,5,--|............      gcd(10,5)
                                     | 5, 0,--|..        gcd(5,0)
                                     | 5, 0,--|..        gcd(5,0) ret 5
                            |10,5, 5|............      gcd(10,5) ret 5
                   |15,10, 5|.....................    gcd(15,10) ret 5
          |25,15, 5|.............................  gcd(25,15) ret 5
|40,25, 5|.....................................gcd(40,25) ret 5
```

18.1 Russian Peasant multiplication

```
int product ( int m, int n ) // assumed nonnegative
{
  if ( n == 0 )
    return 0;
  else
  {
    int p = product ( m, n/2 );
    if ( n%2 == 0 )
    { return p + p; }
    else
    { return p + p + m; }
```

```
  }
}
```

Simulate `product(5,10)`.