12. Desenvolver um programa iterativo e um programa recursivo, equivalente ao programa monolítico apresentado a seguir.

Programa Monolítico – Instruções Rotuladas

```
R01: Faça ler(n) vá_para R02;
R02: Se n <= 0 então vá_para R03 senão vá_para R04;
R03: Faça escrever(erro) vá_para Rx;
R04: Faça h = 0 vá_para R05;
R05: Faça f = 1 vá_para R06;
R06: Faça k = 0 vá_para R07;
R07: Faça s = 1 vá_para R08;
R08: Se n > k então vá_para R09 senão vá_para R18;
R09: Faça k = k + 1 vá_para R10;
R10: Faça p = 1 vá_para R11;
R11: Faça j = k vá_para R12;
R12: Se j > 0 então vá_para R13 senão vá_para R15;
R13: Faça p = p * k vá_para R14;
R14: Faça j = j - 1 vá_para R12;
R15: Faça f = f * k vá_para R16;
R16: Faça h = h + p / f * s vá_para R17;
R17: Faça s = s * (-1) vá_para R08;
R18: Faça escrever(h) vá_para Rx;
```

Programa Iterativo

```
programa
  ler(n);
  se (n <= 0) então
    escrever(error);
  senão
    h = 0;
    f = 1;
    k = 0;
    s = 1;
    enquanto (n > k) faça
      k = k + 1;
      p = 1;
      j = k;
      enquanto (j > 0) faça
        p = p * k;
        j = j - 1;
      fim enquanto;
      f = f * k;
      h = h + p / f * s
      s = s * (-1)
    fim enquanto;
    escrever(h);
  fim se;
fim programa.
```

Programa Recursivo

```
função sub1(j, k)
  se (j > 0)
    então retornar sub1(j - 1, k) * k;
    senão retornar 1;
  fim se;
fim função;

função sub2(n, k, f, s)
  se (n > k) então
    k = k + 1;
    f = f * k;
    retornar sub2(n, k, f, s * (-1)) + sub1(k, k) / f * s;
  senão
    retornar 0;
  fim se;
fim função;


função principal
  ler(n);
  se (n <= 0) então
    escrever(error);
  senão
    escrever(sub2(n, 0, 1, 1));
  fim se;
fim função;
```