```
> restart;
>
```

Maple 12

The procedure W3j is derived from the C program described by William J. Thompson.
William J. Thompson, Angular Momentum: An Illustrated Guide to Rotational Symmetries for Physical Systems, John Wiley and Sons Inc., 1994

Addition of Three Angular Momenta

Using Racah's formula to determine Wigner 3j coeficients

$$\begin{bmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{bmatrix} = (-1)^{j_1 - j_2 - m_3} \cdot \mathrm{N} \cdot S$$

$$\mathrm{N} = \sqrt{\frac{(j3+j1-j2)!(j3-j1+j2)!(j1+j2-j3)!(j3-m3)!(j3+m3)!}{(j1+j2+j3+1)!(j1-m1)!(j1+m1)!(j2-m2)!(j2+m2)!}}$$

$$\mathrm{S} = \sum_{kmin}^{kmax} \frac{(-1)^{(kmin+j2+m2)}(j2+j3+m1-k)!(j1-m1+k)!}{(k)!(j3-j1+j2-k)!(j3-m3-k)!(k+j1-j2+m3)!}$$

where
kmin = maximun of [0 **and** $j2 - j1 - m3$] **and** kmax = minimum of [$j3 - j1 + j2$ **and** $j3 - m3$]

Clebsch-Gordan, C-G, Coefficients from Wigner 3j Coefficients

$$\langle j_1,j_2;m_1,m_2|j_1,j_2;j_3,m_3\rangle = (-1)^{-j_1 + j_2 - m_3}\sqrt{2j_3+1}\begin{bmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & -m_3 \end{bmatrix}$$

**** **Note that we are using the following conventions** ****

coefficient	$\{a,b,c,m_a,m_b,m_c\} \equiv \langle a,b;m_a,m_b|a,b;c,m_c\rangle$
C-G value	$[\pm N] \equiv \pm\sqrt{N}$

This procedure determines the Wigner 3j coefficient using Racah's formula

```
> W3j :=proc(j1, j2, j3, m1, m2, m3)
        local k, kmin, kmax, N, S, P;

        kmin := max(0, j2 − j1 − m3);
        kmax := min( j3 − j1 + j2, j3 − m3);
        P := (-1)^(j1 − j2 − m3);
```

$$N := \sqrt{\frac{(j3+j1-j2)!(j3-j1+j2)!(j1+j2-j3)!(j3-m3)!(j3+m3)!}{(j1+j2+j3+1)!(j1-m1)!(j1+m1)!(j2-m2)!(j2+m2)!}};$$

$$S := \sum_{k=kmin}^{kmax} \frac{(-1)^{(k+j2+m2)}\cdot(j2+j3+m1-k)!(j1-m1+k)!}{(k)!(j3-j1+j2-k)!(j3-m3-k)!(k+j1-j2+m3)!};$$

```
        return (P·N·S);
    end proc:
```

This procedure returns a signed squared coefficient.

$$\pm a\sqrt{n} = \pm a^2 n = \pm \mathbf{N}$$

```
> SQ :=proc(n)
        local sign;
        if (evalf(n) < 0) then sign := -1 else sign := 1 end if;
        return (sign·simplify((n·n)));
    end proc:
```

This procedure determines the value of the C-G coefficients; e.g., ⟨j1,j2;m1,m2|j1,j2;j12,m12⟩

⟨j1,j2;m1,m2|j1,j2;j12,m12⟩ = [± N]

[± N] $\equiv \pm\sqrt{\mathbf{N}}$

```
> VCo :=proc(j1, j2, j3, j13, J, M)
        local m1, m2, m3, m13, c1, c2, s;
        s := " = ";
        for m1 from -j1 by 1 to j1 do
          for m3 from -j3 by 1 to j3 do
            for m13 from -j13 by 1 to j13 do
              for m2 from -j2 by 1 to j2 do
                if (m1 + m3) = m13 and (m13 + m2) = M then    # selection rule
                    c1 := (-1)^(-j1 + j3 − m13)·sqrt(2·j13 + 1) · W3j(j1, j3, j13, m1, m3, -m13);
                    c2 := (-1)^(-j13 + j2 − M)·sqrt(2·J + 1) · W3j(j13, j2, J, m13, m2, -M);
                    printf("      %s[%a]", s, SQ(c1·c2) );
                    printf("|%a,%a,%a;%a,%a,%a> \n", j1, j3, j2, m1, m3, m2);
                    s := " + ";
                 end if;
              end do;
            end do;
          end do;
        end do;
        print( );
    end proc:
```

This procedure returns the product of the CG coefficients and the state |j1,j3,j2;m1,m3,m2⟩

{j1,j3,j13;m1,m3,m13}{j13,j2,J,m13,m2,M} |j1,j3,j2:m1,m3,m2⟩

where j13 = j1 + j3 and J = j13 + j2
m13 = {j13, j13 - 1, ... , − j13} = m1 + m3
M = { J, J - 1, ... , − J} = m13 + m2

```
> SCo := proc(j1, j2, j3, j13, J, M)
      local m1, m2, m3, m13, c, s;
      s := " = ";
      printf("|%a,%a,%a;%a,%a,%a>\n", j1, j3, j2, j13, J, M);
      for m1 from -j1 by 1 to j1 do
       for m3 from -j3 by 1 to j3 do
        for m13 from -j13 by 1 to j13 do
         for m2 from -j2 by 1 to j2 do
          if (m1 + m3) = m13 and (m13 + m2) = M then   # selection rule
           printf("       %s{%a,%a,%a,%a,%a,%a}", s, j1, j3, j13, m1, m3, m13 );
           printf("{%a,%a,%a,%a,%a,%a}", j13, j2, J, m13, m2, M);
           printf("|%a,%a,%a;%a,%a,%a> \n", j1, j3, j2, m1, m3, m2);
           s := " + ";
          end if;
         end do;
        end do;
       end do;
      end do;
     end proc:
```

Main procedure. Add3j adds three angular momenta - Add2j(j1,j2,j3)
using the coupling scheme j1 + j3 = j13 and j13 + j2 = J

$$|j1 - j3| \le j13 \le (j1+j3)$$
$$m13 = \{ j13, j13 - 1, j13 - 2, \ldots , -j13\}$$

$$|j13 - j2| \le J \le (j13 + j2)$$
$$M = \{ J, J - 1, J - 2, \ldots , -J\}$$

```
> Add3j := proc(j1, j2, j3)
       local c, j13, J, M;
       printf("\n    There are %a |j1,j3,j2;j13,J,M> states where,\n\n", (2 j1 + 1)·(2 j2 + 1)·(2 j3
   + 1))
       print(" |j1,j3,j2;j13,J,M> = Σ{j1,j3,j13,m1,m3,m13}{j13,j2,J,m13,m2,M}|j1,j3,j2,m1,m3,m2>
  ");
        c := 0 :  # counter
       for j13 from (j1 + j3) by -1 to |j1 − j3| do
         for J from (j13 + j2) by -1 to |j2 − j13| do
           for M from J by -1 to -J do
             c := c + 1 : printf("%a.", c);
            SCo( j1, j2, j3, j13, J, M);
            VCo( j1, j2, j3, j13, J, M);
           end do;
         end do;
       end do;
     end proc:
```

Example:

> $Add3j\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right);$

```
There are 8 |j1,j3,j2;j13,J,M> states where,

     " |j1,j3,j2;j13,J,M> = Σ{j1,j3,j13,m1,m3,m13}{j13,j2,J,m13,m2,M}|j1,j3,j2,m1,m3,m2> "
1.|1/2,1/2,1/2;1,3/2,3/2>
         = {1/2,1/2,1,1/2,1/2,1}{1,1/2,3/2,1,1/2,3/2}|1/2,1/2,1/2;1/2,1/2,1/2>
         = [1]|1/2,1/2,1/2;1/2,1/2,1/2>

2.|1/2,1/2,1/2;1,3/2,1/2>
         = {1/2,1/2,1,-1/2,1/2,0}{1,1/2,3/2,0,1/2,1/2}|1/2,1/2,1/2;-1/2,1/2,1/2>
         + {1/2,1/2,1,1/2,-1/2,0}{1,1/2,3/2,0,1/2,1/2}|1/2,1/2,1/2;1/2,-1/2,1/2>
         + {1/2,1/2,1,1/2,1/2,1}{1,1/2,3/2,1,-1/2,1/2}|1/2,1/2,1/2;1/2,1/2,-1/2>
         = [1/3]|1/2,1/2,1/2;-1/2,1/2,1/2>
         + [1/3]|1/2,1/2,1/2;1/2,-1/2,1/2>
         + [1/3]|1/2,1/2,1/2;1/2,1/2,-1/2>

3.|1/2,1/2,1/2;1,3/2,-1/2>
         = {1/2,1/2,1,-1/2,-1/2,-1}{1,1/2,3/2,-1,1/2,-1/2}|1/2,1/2,1/2;-1/2,-1/2,1/2>
         + {1/2,1/2,1,-1/2,1/2,0}{1,1/2,3/2,0,-1/2,-1/2}|1/2,1/2,1/2;-1/2,1/2,-1/2>
         + {1/2,1/2,1,1/2,-1/2,0}{1,1/2,3/2,0,-1/2,-1/2}|1/2,1/2,1/2;1/2,-1/2,-1/2>
         = [1/3]|1/2,1/2,1/2;-1/2,-1/2,1/2>
         + [1/3]|1/2,1/2,1/2;-1/2,1/2,-1/2>
         + [1/3]|1/2,1/2,1/2;1/2,-1/2,-1/2>

4.|1/2,1/2,1/2;1,3/2,-3/2>
         = {1/2,1/2,1,-1/2,-1/2,-1}{1,1/2,3/2,-1,-1/2,-3/2}|1/2,1/2,1/2;-1/2,-1/2,-1/2>
         = [1]|1/2,1/2,1/2;-1/2,-1/2,-1/2>

5.|1/2,1/2,1/2;1,1/2,1/2>
         = {1/2,1/2,1,-1/2,1/2,0}{1,1/2,1/2,0,1/2,1/2}|1/2,1/2,1/2;-1/2,1/2,1/2>
         + {1/2,1/2,1,1/2,-1/2,0}{1,1/2,1/2,0,1/2,1/2}|1/2,1/2,1/2;1/2,-1/2,1/2>
         + {1/2,1/2,1,1/2,1/2,1}{1,1/2,1/2,1,-1/2,1/2}|1/2,1/2,1/2;1/2,1/2,-1/2>
         = [-1/6]|1/2,1/2,1/2;-1/2,1/2,1/2>
         + [-1/6]|1/2,1/2,1/2;1/2,-1/2,1/2>
         + [2/3]|1/2,1/2,1/2;1/2,1/2,-1/2>

6.|1/2,1/2,1/2;1,1/2,-1/2>
         = {1/2,1/2,1,-1/2,-1/2,-1}{1,1/2,1/2,-1,1/2,-1/2}|1/2,1/2,1/2;-1/2,-1/2,1/2>
         + {1/2,1/2,1,-1/2,1/2,0}{1,1/2,1/2,0,-1/2,-1/2}|1/2,1/2,1/2;-1/2,1/2,-1/2>
         + {1/2,1/2,1,1/2,-1/2,0}{1,1/2,1/2,0,-1/2,-1/2}|1/2,1/2,1/2;1/2,-1/2,-1/2>
         = [-2/3]|1/2,1/2,1/2;-1/2,-1/2,1/2>
         + [1/6]|1/2,1/2,1/2;-1/2,1/2,-1/2>
         + [1/6]|1/2,1/2,1/2;1/2,-1/2,-1/2>
```

```
7.|1/2,1/2,1/2;0,1/2,1/2>
         = {1/2,1/2,0,-1/2,1/2,0}{0,1/2,1/2,0,1/2,1/2}|1/2,1/2,1/2;-1/2,1/2,1/2>
         + {1/2,1/2,0,1/2,-1/2,0}{0,1/2,1/2,0,1/2,1/2}|1/2,1/2,1/2;1/2,-1/2,1/2>
         = [-1/2]|1/2,1/2,1/2;-1/2,1/2,1/2>
         + [1/2]|1/2,1/2,1/2;1/2,-1/2,1/2>

8.|1/2,1/2,1/2;0,1/2,-1/2>
         = {1/2,1/2,0,-1/2,1/2,0}{0,1/2,1/2,0,-1/2,-1/2}|1/2,1/2,1/2;-1/2,1/2,-1/2>
         + {1/2,1/2,0,1/2,-1/2,0}{0,1/2,1/2,0,-1/2,-1/2}|1/2,1/2,1/2;1/2,-1/2,-1/2>
         = [-1/2]|1/2,1/2,1/2;-1/2,1/2,-1/2>
         + [1/2]|1/2,1/2,1/2;1/2,-1/2,-1/2>
```

(1)

```
>
```