Advanced Programs in SymbolicC++

by

Willi-Hans Steeb International School for Scientific Computing at University of Johannesburg, South Africa

and

Yorick Hardy Department of Mathematical Sciences at University of South Africa, South Africa

Text Book

Computer Algebra with SymbolicC++

by Yorick Hardy, Kiat Shi Tan and Willi-Hans Steeb World Scientific Publishing, Singapore 2008 ISBN 981 256 916 2 http://www.worldscibooks.com/mathematics/6966.html

Problem 1. Bernoulli Numbers and Euler Summation Formula

The Bernoulli numbers B_0, B_1, B_2, \ldots are defined by the power series expansion

$$\frac{x}{e^x - 1} = \sum_{j=0}^{\infty} \frac{B_j}{j!} x^j \equiv B_0 + \frac{B_1}{1!} x + \frac{B_2}{2!} x^2 + \cdots$$

One finds $B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, $B_3 = 0$, $B_4 = -1/30$. One has $B_j = 0$ if $j \geq 3$ and j odd. We utilize a Maclaurin series and the L'Hospital rule (since $x \to 0$) to evaluate the Bernoulli numbers. This is obviously not the most effective way to find the Bernoulli numbers. The Bernoulli numbers are utilized in the Euler summation formula

$$\sum_{j=1}^{n} f(j) = \int_{1}^{n} f(t)dt + \frac{1}{2}(f(n) + f(1)) + \sum_{k=1}^{n} \frac{B_{2k}}{(2k)!} (f^{(2k-1)}(n) - f^{(2k-1)}(1)) + R_m(n)$$

where

$$|R_m(n)| \le \frac{4}{(2\pi)^{2m}} \int_1^m |f^{(2m)}(t)| dt.$$

We apply it to

$$\sum_{i=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}.$$

Solution 1.

```
// bernoulli.cpp

#include <iostream>
#include "symbolicc++.h"

using namespace std;

Symbolic Bernoulli(int n)
{
  int i;
  Symbolic x("x"), fn, fd, fn0, fd0, lhfn, lhfd;
  Symbolic B("B", n+1);
  fn = x; fd = exp(x) - 1;
```

```
for(i=0;i<=n;++i)
 lhfn = fn; lhfd = fd;
 do
 {
 fn0 = lhfn[x==0]; fd0 = lhfd[x==0];
 lhfn = df(lhfn, x); lhfd = df(lhfd, x);
} while(fn0==0 && fd0==0);
B(i) = fn0/fd0;
fn = df(fn,x)*fd - fn*df(fd,x);
fd *= fd;
}
return B;
}
int fact(int n)
int f = 1;
while(n > 0) f *= n--;
return f;
}
int main(void)
{
int n = 3, m = 2, k;
Symbolic B = Bernoulli(2*m);
 Symbolic j("j"), f = j^2;
Symbolic sum = integrate(f,j,1,n) + (f[j==1]+f[j==n])/2;
for(k=1;k<=m;k++)
Symbolic Df = df(f,j,2*k-1);
sum += B(int(2*k))*(Df[j==n]-Df[j==1])/fact(2*k);
}
cout << sum << endl;</pre>
return 0;
}
```

Problem 2. Lie Algebras and Adjoint Representation

Let L be a Lie algebra. One analyzes the structure of a Lie algebra L by studying its ideals. A subspace I of a Lie algebra L is called an *ideal* of L if $x \in L$ and $y \in I$ together imply $[x, y] \in I$. If L has no ideals except itself and 0, and if moreover $[L, L] \neq 0$ we call the Lie algebra L simple.

If L is a Lie algebra and $x \in L$, the operator adX that maps y to [x, y] is a linear transformation of L into itself

$$(\mathrm{ad}x)y := [x, y].$$

Then $x \to adx$ is a representation of the Lie algebra L with L itself considered as the vector space of the representation.

The Killing form of a Lie algebra is the symmetric bilinear form

$$K(x,y) := \operatorname{tr}(\operatorname{ad}x\operatorname{ad}y)$$

where tr denotes the trace. If x_j (j = 0, 1, ..., n - 1) form a basis of L then

$$g_{jk} = K(x_j, x_k)$$

is called the *metric tensor* for L. Note that a Lie algebra is semisimple if and only if the matrix (g_{jk}) is nonsingular, i.e., $\det((g_{jk})) \neq 0$.

We consider the simple Lie algebra so(3) with the basis elements x(0), x(1), x(2) and the commutation relations

$$[x(0), x(1)] = x(2),$$
 $[x(1), x(2)] = x(0),$ $[x(2), x(0)] = x(1).$

We find

$$adx(0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad adx(1) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad adx(2) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Solution 2.

```
// adjoint.cpp
#include <iostream>
#include "symbolicc++.h"
using namespace std;
int main(void)
{
  int n=3;
  Symbolic X("X",n);
  Symbolic comm("",n,n);
  comm(0,0) = Symbolic(0), comm(0,1) = X(2); comm(0,2) = -X(1);
  comm(1,0) = -X(2); comm(1,1) = Symbolic(0); comm(1,2) = X(0);
  comm(2,0) = X(1); comm(2,1) = -X(0); comm(2,2) = Symbolic(0);
  Symbolic adX("adX",n);
  int i,j,k;
  for(i=0;i<n;i++)
   Symbolic rep("",n,n);
   for(j=0;j<n;j++)
   for(k=0;k<n;k++) rep(j,k) = comm(j,k).coeff(X(i));
   adX(i) = rep;
   cout << "adX(" << i << ") = " << adX(i) << endl;</pre>
  }
  Symbolic g("",n,n);
  for(j=0;j<n;j++)
   for(k=0; k< n; k++) g(j,k) = tr(adX(j)*adX(k));
  cout << "g = " << g << endl;</pre>
 return 0;
}
```

Problem 3. Number Theory

Any positive integer n can be written uniquely as

$$n = 2^j + k$$

where $0 \le k < 2^j$. For example $10 = 2^3 + 2$. We write a C++ program that finds j

and k for a given n and use templates so that also the Verylong class of SymbolicC++ can be used.

Solution 3.

```
// integerrep.cpp
#include <iostream>
#include "verylong.h"
using namespace std;
template <class T>
void intrep(const T &n,T &j,T &k)
{
T zero = T();
T one = zero;
T two = zero;
T N = n;
++one; ++two; ++two;
j = 0;
T \exp 2j = one; // 2^j, 2^0 = 1
while(N != one) { ++j; exp2j *= two; N /= two; }
k = n-exp2j;
}
int main(void)
{
int x = 7932467, xj, xk;
Verylong y("21478658832956723145782"), yj, yk;
intrep(x,xj,xk);
cout << x << " = " << "2^" << xj << " + " << xk << endl;
 intrep(y,yj,yk);
cout << y << " = " << "2^" << yj << " + " << yk << endl;
return 0;
}
```

Problem 4. Lie Groups

Consider the Lie group SU(2) which consists of all unitary matrices $U = (u_{k\ell})_{1 \le k, \ell \le 2}$ with $\det(U) = 1$. For example, the Pauli spin matrices are not elements of SU(2), but $i\sigma_x$, $i\sigma_y$, $i\sigma_z$ are. Let

$$j \in \{1/2, 1, 3/2, 2, \ldots\}.$$

The unitary irreducible representations of the Lie group SU(2) are given by

$$\begin{split} D^{j}_{m,m'}(U) &= \sqrt{(j+m)!(j-m)!(j+m')!(j-m')!} \\ &\times \sum_{k} \frac{u^{k}_{11}u^{j+m-k}_{12}u^{j+m'-k}_{21}u^{k-m-m'}_{22}}{k!(j+m-k)!(j+m'-k)!(k-m-m')!} \end{split}$$

where m, m' take the values $-j, -j + 1, \ldots, j$. The sum k runs over all nonnegative values of k for which all factorials in the denominator are nonnegative. Give an implementation in SymbolicC++ and apply it to the matrices $i\sigma_x$, $i\sigma_y$, $i\sigma_z$,

$$iU_H = \frac{1}{\sqrt{2}} \begin{pmatrix} i & i \\ i & -i \end{pmatrix}.$$

Solution 4. We set $u_{11} = u11$, $u_{12} = u12$, $u_{21} = u22$, $I_2 = I2$.

```
// rep.cpp
#include <iostream>
#include "symbolicc++.h"
using namespace std;
int main(void)
{
    Symbolic j, m, mp;
    Symbolic u11("u11"), u12("u12"), u21("u21"), u22("u22");
    using SymbolicConstant::i;
    int k, kl, ku;
    for(j=1/Symbolic(2);j!=2;j+=1/Symbolic(2))
    {
        cout << "j = " << j << endl;
        Symbolic Dj("", int(2*j+1), int(2*j+1));</pre>
```

```
for(m=-j;int(m+j)<=int(2*j);m++)</pre>
  for(mp=-j;int(mp+j)<=int(2*j);mp++)</pre>
  {
  Symbolic sq
    = sqrt(factorial(j+m)*factorial(j-m))*factorial(j+mp)*factorial(j-mp));
  Symbolic sum = 0;
  kl = (int(m+mp) > 0) ? int(m+mp) : 0;
  ku = (int(j+m) > int(j+mp)) ? int(j+mp) : int(j+m);
  for(k=k1; k<=ku; k++)
   sum += (u11^k)*(u12^(j+m-k))*(u21^(j+mp-k))*(u22^(k-m-mp))
        /(factorial(k)*factorial(j+m-k)*factorial(j+mp-k)*factorial(k-m-mp));
 Dj(int(j-m), int(j-mp)) = sq*sum;
 }
 Dj = Dj[sqrt(Symbolic(4))==2,
          sqrt(Symbolic(12))==2*sqrt(Symbolic(3)),
          sqrt(Symbolic(36))==6];
  cout << Dj << endl;</pre>
  cout << "For u=I2:" << endl</pre>
       << Dj[u11==1,u12==0,u21==0,u22==1] << endl;
  cout << "For u=i sigma x:" << endl</pre>
       << Dj[u11==0,u12==i,u21==i,u22==0] << endl;
  cout << "For u = i sigma y:" << endl</pre>
       << Dj[u11==0,u12==1,u21==-1,u22==0] << endl;
  cout << "For u = i sigma z:" << endl</pre>
       << Dj[u11==i,u12==0,u21==0,u22==-i] << endl;
  cout << "For u = i U_H:" << endl</pre>
       << Dj[u11==i/sqrt(Symbolic(2)),u12==i/sqrt(Symbolic(2)),</pre>
             u21==i/sqrt(Symbolic(2)),u22==-i/sqrt(Symbolic(2))] << endl;</pre>
 }
return 0;
}
The output is
j = 1/2
[u11 u12]
```

```
[u21 u22]
For u = I2:
[1 0]
[0 1]
For u = i sigma x:
[0 i]
[i 0]
For u = i sigma y:
[ 0 1]
[-1 0]
For u = i \text{ sigma } z:
[i 0]
[ 0 -i]
For u = i U_H:
[ i*(2)^(-1/2) i*(2)^(-1/2)]
[i*(2)^(-1/2) -i*(2)^(-1/2)]
j = 1
[ u11^{(2)} (2)^{(1/2)}*u11*u12 u12^{(2)}
                                                      ]
 [(2)^{(1/2)}*u11*u21 \quad u12*u21+u11*u22 \quad (2)^{(1/2)}*u12*u22] 
[
    u21^(2)
                (2)^(1/2)*u21*u22
                                        u22^(2)
                                                      ]
For u = 12:
[1 0 0]
[0 1 0]
[0 0 1]
For u = i sigma x:
[ 0 0 -1]
[ 0 -1 0]
[-1 0 0]
For u = i sigma y:
[0 0 1]
[ 0 -1 0]
[1 0 0]
For u = i sigma z:
[-1 \ 0 \ 0]
[ 0 1 0]
[ 0 0 -1]
```

```
For u = i U_H:
[ -1/2
            -(2)^{(-1/2)}
                             -1/2 ]
[-(2)^{(-1/2)}
                          (2)^{(-1/2)}
                0
              (2)^{(-1/2)}
                             -1/2 ]
  -1/2
j = 3/2
u11^(3)
                             (3)^(1/2)*u11^(2)*u12
                                                        (3)^{(1/2)}*u11*u12^{(2)}
                                                                                         u12
[ (3)^{(1/2)}*u11^{(2)}*u21
                           2*u11*u12*u21+u11^(2)*u22 u12^(2)*u21+2*u11*u12*u22
                                                                                  (3)^(1/2)*
[ (3)^(1/2)*u11*u21^(2)
                           u12*u21^(2)+2*u11*u21*u22 2*u12*u21*u22+u11*u22^(2)
                                                                                   (3)^(1/2)*
          u21^(3)
                                                        (3)^(1/2)*u21*u22^(2)
                             (3)^{(1/2)}*u21^{(2)}*u22
                                                                                         u22
For u = I2:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
For u = i \text{ sigma } x:
Γ
      0
                                  (-1)^(3/2)
                            0
Γ
      0
                           -i
                                       0
                                            ]
[
      0
                -i
                            0
                                       0
                                            ]
[(-1)^{(3/2)}
                            0
                                            ]
                                       0
For u = i \text{ sigma } y:
[0001]
[ 0 0 -1 0]
[ 0 1 0 0]
[-1 0 0 0]
For u = i sigma z:
[(-1)^{(3/2)}
                                                ]
      0
                              0
                                           0
                                                 ]
Γ
                  0
                                                 1
      0
                              -i
                                           0
0
                  0
                              0
                                     -(-1)^(3/2)]
For u = i U_H:
[ (-1)^{(3/2)*(2)^{(-3/2)}}
                             -1/2*(3)^(1/2)*i*(2)^(-1/2) -1/2*(3)^(1/2)*i*(2)^(-1/2)
                                                                                         (-1
[-1/2*(3)^(1/2)*i*(2)^(-1/2)
                                  -1/2*i*(2)^(-1/2)
                                                                1/2*i*(2)^(-1/2)
                                                                                       1/2*(
[-1/2*(3)^(1/2)*i*(2)^(-1/2)
                                   1/2*i*(2)^(-1/2)
                                                                1/2*i*(2)^(-1/2)
                                                                                      -1/2*(
```

 $1/2*(3)^(1/2)*i*(2)^(-1/2) -1/2*(3)^(1/2)*i*(2)^(-1/2)$

-(-

Problem 5. Bose operators and adjoint representation

 $[(-1)^(3/2)*(2)^(-3/2)$

Let b_1^{\dagger} , b_2^{\dagger} be Bose creation operator and let I be the identity operator. The semisimple Lie algebra su(1,1) is generated by

$$K_{+} := b_{1}^{\dagger} b_{2}^{\dagger}, \quad K_{-} := b_{1} b_{2}, \quad K_{0} := \frac{1}{2} (b_{1}^{\dagger} b_{1} + b_{2}^{\dagger} b_{2} + I)$$

with the commutation relations

$$[K_0, K_+] = K_+, \quad [K_0, K_-] = -K_-, \quad [K_-, K_+] = 2K_0.$$

We use the ordering K_+ , K_- , K_0 for the basis. We find the *ajoint representation* for the Lie algebra.

Solution 5. Using the ordering given above for the basis we have

$$(adK_{+})K_{+} = [K_{+}, K_{+}] = 0$$

 $(adK_{+})K_{-} = [K_{+}, K_{-}] = -2K_{0}$
 $(adK_{+})K_{0} = [K_{+}, K_{0}] = -K_{+}.$

Thus

$$(K_{+} \quad K_{-} \quad K_{0}) \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & -2 & 0 \end{pmatrix} = (0 \quad -2K_{0} \quad -K_{+}).$$

Since K_+ , K_- , K_0 is a basis of the Lie algebra it follows that

$$adK_{+} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & -2 & 0 \end{pmatrix}.$$

Analogously we find

$$adK_{-} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 2 & 0 & 0 \end{pmatrix}, \quad adK_{0} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

With $X(0) = K_+$, $X(1) = K_-$, $X(2) = K_0$ the SymbolicC++ implementation is:

// adjointrep.cpp

```
// g++ -I . adjoint.cpp -o adjoint
#include <iostream>
#include "symbolicc++.h"
using namespace std;
int main(void)
{
 int i, j, k;
 const int n=3;
 Symbolic ad[n];
for(j=0;j<n;j++) ad[j] = Symbolic("",n,n);</pre>
 Symbolic X("X",n);
 Symbolic Y("Y",n,n);
Y(0,0) = 0; Y(0,1) = -2*X(2); Y(0,2) = -X(0);
Y(1,0) = 2*X(2); Y(1,1) = 0; Y(1,2) = X(1);
Y(2,0) = X(0); Y(2,1) = -X(1); Y(2,2) = 0;
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
   for(k=0; k< n; k++) ad[i](j,k) = Y(i,k).coeff(X(j));
for(j=0; j< n; j++) cout << "ad[" << j << "]"
                        << "=" << ad[j] << endl;
return 0;
}
```

Problem 6. de Sitter Group

The non-compact de Sitter group SO(1,4) consists of all real 5×5 matrices such that the quadratic form

$$x_0^2 - x_1^2 - x_2^2 - x_3^2 - x_4^2$$

is invariant. The corresponding Lie algebra so(1,4) consists of all real matrices of the

form

$$L = \begin{pmatrix} 0 & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{01} & 0 & -a_{12} & -a_{13} & -a_{14} \\ a_{02} & a_{12} & 0 & -a_{23} & -a_{24} \\ a_{03} & a_{13} & a_{23} & 0 & -a_{34} \\ a_{04} & a_{14} & a_{24} & a_{34} & 0 \end{pmatrix}.$$

The dimension of so(1,4) is 10. The semi-simple Lie algebra so(1,4) has the basis elements

$$L_{rs} = -E_{rs} + E_{sr}, \quad s, r = 1, 2, 3, 4, \ s < r$$

 $L_{0r} = E_{0r} + E_{r0}, \quad r = 1, 2, 3, 4$

where E_{rs} is the matrix with entries $(E_{rs})_{pq} = \delta_{rp}\delta_{sq}$ and δ_{rp} denotes the Kronecker delta. The commutation relations are

$$[L_{\mu\nu}, L_{\rho\sigma}] = g_{\nu\rho}L_{\mu\sigma} + g_{\mu\sigma}L_{\nu\rho} - g_{\mu\rho}L_{\nu\sigma} - g_{\nu\sigma}L_{\mu\rho}$$

where $\rho, \mu, \nu, \sigma = 0, 1, 2, 3, 4$ and $g_{k0} = g_{0k} = \delta_{0k}$, $g_{ks} = -\delta_{ks}$, k, s = 1, 2, 3, 4. An element of SO(1, 4) is given by

$$S(\alpha) = \begin{pmatrix} \cosh(\alpha) & 0 & 0 & 0 & \sinh(\alpha) \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \sinh(\alpha) & 0 & 0 & 0 & \cosh(\alpha) \end{pmatrix}.$$

Find the generator

$$G = \left. \frac{dS(\alpha)}{d\alpha} \right|_{\alpha=0}$$

using SymbolicC++.

Solution 6. In the SymbolicC++ program the generator G is calculated (i.e. $S(\alpha) = \exp(\alpha G)$. Thus G is an element of the Lie algebra so(1,4).

```
// deSitter.cpp
// g++ -I . deSitter.cpp -o deSitter -fno-elide-constructor
#include <iostream>
#include "symbolicc++.h"
```

```
using namespace std;
int main(void)
{
 const int n=5;
 Symbolic alpha("alpha");
Symbolic X("X",n,n);
 Symbolic dX("dX",n,n);
Symbolic result;
X(0,0) = \cosh(alpha); X(0,1) = 0; X(0,2) = 0; X(0,3) = 0; X(0,4) = \sinh(alpha);
X(1,0) = 0; X(1,1) = 1; X(1,2) = 0; X(1,3) = 0; X(1,4) = 0;
X(2,0) = 0; X(2,1) = 0; X(2,2) = 1; X(2,3) = 0; X(2,4) = 0;
X(3,0) = 0; X(3,1) = 0; X(3,2) = 0; X(3,3) = 1; X(3,4) = 0;
X(4,0) = \sinh(alpha); X(4,1) = 0; X(4,2) = 0; X(4,3) = 0; X(4,4) = \cosh(alpha);
cout << X << endl;</pre>
dX = df(X,alpha); cout << dX << endl;</pre>
dX = dX[alpha==0]; cout << dX << endl;</pre>
return 0;
}
```

The output is

Problem 7. Temperley Lieb Algebra

Let $n \geq 2$ and $j, k = 1, \ldots, 2n - 1$. Let U be the 4×4 matrix

$$U = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & s^4 & 1 & 0 \\ 0 & 1 & s^{-4} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then

$$U^2 = (s^4 + s^{-4})U.$$

Consider the 2^{2n} matrices

$$U_i = I_2 \otimes \cdots \otimes I_2 \otimes U \otimes I_2 \otimes \cdots \otimes I_2$$

where the 4×4 matrix U is at the jth position. These matrices satisfy the relations (Temperley-Lieb algebra)

$$U_{j}U_{j} = (s^{4} + s^{-4})U_{j}$$

 $U_{j}U_{j\pm 1}U_{j} = U_{j}$
 $U_{j}U_{k} = U_{k}U_{j} \text{ if } |j-k| \ge 2.$

These matrices play a role for the q-state Potts model. Consider the case n=2 and the 16×16 matrices

$$U_1 = U \otimes I_2 \otimes I_2$$
, $U_2 = I_2 \otimes U \otimes I_2$, $U_3 = I_2 \otimes I_2 \otimes U$.

Write computer algebra programs in SymbolicC++ and Maxima which check that these matrices satisfy

$$U_1U_1 = (s^4 + s^{-4})U_1$$
, $U_2U_2 = (s^4 + s^{-4})U_2$, $U_3U_3 = (s^4 + s^{-4})U_3$
 $U_1U_2U_1 = U_1$, $U_2U_3U_2 = U_2$, $U_3U_2U_3 = U_3$, $U_2U_1U_2 = U_2$
 $U_1U_3 = U_3U_1$.

Solution 7. The SymbolicC++ program is

```
U1 = kron(U,kron(I2,I2));
 U2 = kron(I2, kron(U, I2));
 U3 = kron(I2, kron(I2, U));
 R0 = U1*U1-((s^4)+1/(s^4))*U1;
 cout << "R0 = " << R0 << endl;
 R1 = U1 - U1*U2*U1;
 cout << "R1 = " << R1 << endl;
 R2 = U1*U3 - U3*U1;
 cout << "R2 = " << R2 << endl;
 return 0;
}
The Maxima program is
/* Temperley.mac */
I2: matrix([1,0],[0,1]);
U: matrix([0,0,0,0],[0,s^4,1,0],[0,1,1/s^4,0],[0,0,0,0]);
U1: kronecker_product(U,kronecker_product(I2,I2));
U2: kronecker_product(I2,kronecker_product(U,I2));
U3: kronecker_product(I2,kronecker_product(I2,U));
RO: U1 . U1 - (s^4 + 1/s^4)*U1;
RO: expand(RO);
print("R0",R0);
R1: U1 - U1 . U2 . U1;
R1: expand(R1);
print("R1",R1);
R2: U1 . U3 - U3 . U1;
R2: expand(R2);
print("R2",R2);
```

Problem 8. Transitivity Set

Let $P = \{P_0, P_2, \dots, P_{k-1}\}$ be a *semi-group* of permutations on the set $\Omega = \{0, 1, \dots, n-1\}$. A semi-group is closed under the group operation, but the inverse and identity need not be a member of the set. The transitivity set (or orbit)

containing $i \in \Omega$ is the set of images under the action of products of elements of P. Write a C++ program which finds the orbits given P and Ω .

Solution 8. Since P is closed under products of elements of P, we need only consider the actions of lements of P. The image $P_j(i)$ of $i \in \Omega$ under the action of $P_j \in P$ is written as $\text{im}[j][i] = P_j(i)$ in the program. In the program we consider the group of permutations on 3 elements and a subgroup of thr group of permutations on 4 elements.

```
// transitivity.cpp
#include <iostream>
#include <map>
#include <vector>
using namespace std;
void orbits(map<int,map<int,int> > im, map<int,int> &ind,
           map<int, vector<int> > &orb)
{
int k = im.size();
int n = im.begin()->second.size();
 int i, j, orbits = 0, orbit;
vector<bool> hasorbit(n, 0);
for(i=0;i<n;i++)
 if(!hasorbit[i])
                          // this is a new orbit
 {
  orbit = orbits;
  orbits++;
  orb[orbit] = vector<int>(1);
  orb[orbit][0] = i;  // the orbit starts with i
  ind[i] = orbit;
  hasorbit[i] = 1;
  for(j=0;j<k;j++)
   {
   int q = im[j][i];  // find each image of i
   if(!hasorbit[q])
   {
```

```
hasorbit[q] = 1;
     ind[q] = orbit;  // each image is in the same orbit
     orb[orbit].push_back(q);
    }
  }
 }
}
int main(void)
{
int j, k;
map<int, map<int, int> > im;
map<int,int> ind;
map<int, vector<int> > orb;
cout << "Full permutation group." << endl;</pre>
 im[0][0] = 0; im[0][1] = 1; im[0][2] = 2; // identity
 im[1][0] = 0; im[1][1] = 2; im[1][2] = 1;
 im[2][0] = 1; im[2][1] = 0; im[2][2] = 2;
 im[3][0] = 1; im[3][1] = 2; im[3][2] = 0;
 im[4][0] = 2; im[4][1] = 0; im[4][2] = 1;
 im[5][0] = 2; im[5][1] = 1; im[5][2] = 0;
orbits(im,ind,orb);
cout << "Group element orbit indices:" << endl;</pre>
for(j=0;j<int(ind.size());j++)</pre>
 cout << "ind[" << j << "] = " << ind[j] << endl;</pre>
cout << "Orbits:" << endl;</pre>
for(j=0;j<int(orb.size());j++)</pre>
  cout << "orb[" << j << "] = ";
 for(k=0; k<int(orb[j].size()); k++)</pre>
  cout << orb[j][k] << " ";
 cout << endl;</pre>
im.clear();
cout << endl;</pre>
```

```
cout << "Permutation subgroup." << endl;</pre>
 im[0][0] = 0; im[0][1] = 1; im[0][2] = 2; im[0][3] = 3;
 im[1][0] = 1; im[1][1] = 0; im[1][2] = 3; im[1][3] = 2;
 orbits(im,ind,orb);
 cout << "Group element orbit indices:" << endl;</pre>
for(j=0;j<int(ind.size());j++)</pre>
  cout << "ind[" << j << "] = " << ind[j] << endl;</pre>
 cout << "Orbits:" << endl;</pre>
 for(j=0;j<int(orb.size());j++)</pre>
  cout << "orb[" << j << "] = ";
  for(k=0;k<int(orb[j].size());k++)</pre>
   cout << orb[j][k] << " ";
  cout << endl;</pre>
 }
return 0;
}
```

The program output is as follows. For the permutation group on 3 elements there is only one orbit. For the subgroup of the permutation group of 4 elements there are two orbits.

```
Full permutation group.
Group element orbit indices:
ind[0] = 0
ind[1] = 0
ind[2] = 0
Orbits:
orb[0] = 0 1 2
Permutation subgroup.
Group element orbit indices:
ind[0] = 0
ind[1] = 0
ind[2] = 1
ind[3] = 1
Orbits:
```

$$orb[0] = 0 1$$

 $orb[1] = 2 3$

Problem 9. Braid Group

Let $\{\beta_1, \ldots, \beta_n\}$ be the generators of the Braid group B_{n+1} where

$$\beta_i \beta_{i+1} \beta_i = \beta_{i+1} \beta_i \beta_{i+1}, \qquad \beta_i^2 = 1, \qquad \beta_i \beta_j = \beta_j \beta_i$$

where $j, i \in \{1, ..., n\}$ and $|j - i| \neq 1$. Let $\{\mathbf{e}_1, ..., \mathbf{e}_{n+1}\}$ denote the standard basis in \mathbb{C}^{n+1} . The action ρ of an element β of the braid group B_n on a vector $\mathbf{x} = (x_1, ..., x_{n+1}) \in \mathbb{C}^n$ is given by

$$\rho(\beta_j)(x_1\mathbf{e}_1 + \dots + x_{n+1}\mathbf{e}_{n+1})$$

$$= x_1\mathbf{e}_1 + \dots + x_{i-1}\mathbf{e}_{i-1} + x_{i+1}\mathbf{e}_i + (x_i+1)\mathbf{e}_{i+1} + x_{i+2}\mathbf{e}_{i+2} + \dots + x_{n+1}\mathbf{e}_{n+1}.$$

and $\rho(\beta_i\beta_j)\mathbf{x} = \rho(\beta_i)\rho(\beta_j)\mathbf{x}$.

We have the property $\rho((\beta_i\beta_{i+1})^3) = \rho((\beta_{i+1}\beta_i)^3) = T_i$, where

$$T_i(\mathbf{x}) = \mathbf{x} + 2(\mathbf{e}_i + \mathbf{e}_{i+1} + \mathbf{e}_{i+2}).$$

Give a C++ implementation of ρ .

Solution 9. We use SymbolicC++ for the representation. A matrix representation $\tilde{\rho}$ of ρ follows directly if we define $\tilde{\mathbf{x}} = (x_1, \dots, x_{n+1}, 1) \in \mathbb{C}^{n+2}$. Then we have

$$\tilde{\rho}(\beta_i) = I_{i-1} \oplus \sigma_x \oplus I_{n-i+1} + (0) \oplus (\mathbf{e}_i \mathbf{e}_{n+1}^T)$$

where I_j is the $j \times j$ identity matrix. Thus

$$\tilde{\rho}(\beta_j)(x_1,\ldots,x_{n+1},1)^T = (x_1,\ldots,x_{i-1},x_{i+1},x_i+1,x_{i+2},\ldots,x_{n+1},1)^T.$$

We use this matrix representation in the program below.

// braid.cpp

#include <iostream>
#include "symbolicc++.h"
using namespace std;

```
int n = 5;
Symbolic beta = ~Symbolic("beta",n); // non-commutative
Symbolic rho(const Symbolic &b)
static int first = 1;
static Equations representation;
int i;
if(first)
 {
 Symbolic rho;
 for(i=0;i<n;i++)
  rho = Symbolic("",n+2,n+2).identity();
  rho(i,i) = 0;
  rho(i+1,i+1) = 0;
  rho(i,i+1) = 1;
  rho(i+1,i) = 1;
  rho(i+1,n+1) = 1;
  representation = (representation,beta(i)==rho);
 }
 first = 0;
 }
return b.subst_all(representation);
}
Symbolic apply(const Symbolic &rho,const Symbolic &x)
{
int i;
Symbolic y("y",x.rows()+1), z, zr = x;
for(i=0;i<x.rows();i++) y(i) = x(i);
y(x.rows()) = 1;
z = rho*y;
for(i=0; i<x.rows(); i++) zr(i) = z(i);
return zr;
```

```
}
int main(void)
{
 int n = 5;
 Symbolic x("x",n+1), r;
 cout << x << endl;</pre>
 r = apply(rho(beta(2)),x);
 cout << r << endl;</pre>
r = apply(rho(beta(2)*beta(3)*beta(2)*beta(3)*beta(2)*beta(3)),x);
 cout << r << endl;</pre>
 r = apply(rho(beta(3)*beta(2)*beta(3)*beta(2)*beta(3)*beta(2)),x);
 cout << r << endl;</pre>
return 0;
}
The program output is as follows.
[x0]
[x1]
[x2]
[x3]
[x4]
[x5]
[ x0 ]
[ x1 ]
[ x3 ]
[x2+1]
[ x4 ]
[ x5 ]
[ x0 ]
[ x1 ]
[x2+2]
[x3+2]
[x4+2]
[ x5 ]
```

[x0]

[x1]

[x2+2]

[x3+2]

[x4+2]

[x5]

Problem 10. Function Composition

Let A, B be sets. Let f be a function of A into B. We say that f is:

surjective (onto) if f(A) = B

injective (one-one) if for all $x, x' \in A$, $f(x) = f(x') \Rightarrow x = x'$.

bijective (one-one onto) if it is both injective and surjective.

Let $f:A\to B$, $g:B\to C$ be functions of A into B and B into C, respectively. The composite function of f and g $((g\circ f):A\to C)$ is denoted by $g\circ f$ and defined as

$$(g \circ f)(x) = g(f(x))$$
 for all $x \in A$.

Consider the analytic functions $f: \mathbb{R} \to \mathbb{R}, g: \mathbb{R} \to \mathbb{R}$

$$f(x) = x^2 + 1,$$
 $g(x) = 2x + 2.$

Find the function

$$h(x) = f(g(x)) - g(f(x)).$$

using SymbolicC++. Find h(x = 2.5) using the symbolic-numeric interface of SymbolicC++.

Solution 10. We have

$$f(g(x)) = (2x+2)^2 = 4x^2 + 8x + 5$$

$$g(f(x)) = 2(x^2+1) + 2 = 2x^2 + 4$$

$$f(g(x)) - g(f(x)) = 2x^2 + 8x + 1.$$

// fcommutator2.cpp

#include <iostream>

```
#include "symbolicc++.h"
using namespace std;
Symbolic f(const Symbolic &x) { return x*x+1; }
Symbolic g(const Symbolic &x) { return 2*x+2; }
Symbolic C(Symbolic (*f)(const Symbolic&), Symbolic (*g)(const Symbolic&),
           const Symbolic &x)
{
   return (f(g(x))-g(f(x)));
}
int main(void)
{
  Symbolic x("x");
  Symbolic result = C(f,g,x);
  cout << "result = " << result << endl;</pre>
  double v = 2.5;
  double resultvalue = result[x==v];
  cout << "resultvalue = " << resultvalue << endl;</pre>
 return 0;
}
```

Problem 11. Equation, Verylong and Rational

Let $n_1, n_2, n_3 \in \mathbb{N}$. Consider the equation

$$\frac{1}{n_1} + \frac{1}{n_2} + \frac{1}{n_3} = 1.$$

Write a SymbolicC++ program utilizing the class Rational and Verylong to test whether there are solutions for $n_1, n_2, n_3 \in \{1, 2, ..., 50\}$. Note that the equation is symmetric under the permutations of (n_1, n_2, n_3) .

Solution 11. The following program

```
// eqRationalVerylong.cpp
#include <iostream>
#include <string>
```

```
#include "verylong.h"
#include "rational.h"
using namespace std;
int main(void)
{
 Verylong zero("0"); Verylong one("1");
 Verylong fifty("50");
 Verylong n1, n2, n3;
 Rational<Verylong> half("1/2");
 Rational < Verylong > t1, t2, t3, sum;
 for(n1=one;n1<=fifty;n1++)</pre>
    for(n2=one;n2<=fifty;n2++)</pre>
      for(n3=one;n3<=fifty;n3++)
      t1 = (Rational<Verylong>(one))/(Rational<Verylong>(n1));
      t2 = (Rational < Verylong > (one)) / (Rational < Verylong > (n2));
      t3 = (Rational < Verylong > (one))/(Rational < Verylong > (n3));
      sum = t1 + t2 + t3;
      if(sum==half)
      cout << "n1=" << n1 << " " << "n2=" << n2 << " "
           << "n3=" << n3 << endl;
      }
   return 0;
}
the solutions
           (3,7,42), (3,8,24), (3,9,18), (3,10,15), (4,5,20), (4,6,12)
```

together with all the possible permutations.