Garamon cheat sheet (CGA)

Includes

```
#include <c3ga/Mvec.hpp>  // the lib
#include "c3gaTools.hpp"  // some optional tools

using namespace c3ga;
```

Constructor and accessor

```
Mvec<double> mv1;
mv1[scalar] = 1.0;
mv1[E0] = mv1[Ei1] = 42.0;
std::cout << "mv1 : " << mv1 << std::endl;

Mvec<double> mv2 = I<double>() + 2*e01<double>();
std::cout << "mv2 : " << mv2 << std::endl << std::endl;
```

Products

```
std::cout << "outer product     : " << (mv1 ^ mv2) << std::endl;
std::cout << "inner product     : " << (mv1 | mv2) << std::endl;
std::cout << "geometric product : " << (mv1 * mv2) << std::endl;
std::cout << "left contraction  : " << (mv1 < mv2) << std::endl;
std::cout << "right contraction : " << (mv1 > mv2) << std::endl;
std::cout << "divide            : " << (mv1 / mv2) << std::endl;
std::cout << "inverse           : " <<  mv1.inv()  << std::endl;
```

GA Utilities

```
std::cout << "dual  : " << !mv1 << std::endl; // or mv1.dual()
std::cout << "grade : " << mv1.grade() << std::endl;
std::cout << "norm  : " << mv1.norm()  << std::endl;
mv1.clear();
if(mv1.isEmpty()) std::cout << "mv1 is empty: ok" << std::endl;
```

c3gaTools (optional)

```
Mvec<double> mv1 = point<double>(-2,0,0.5);
Mvec<double> mv2 = randomPoint<double>();  // components ranging in [-1,1]
Mvec<double> mv3 = dualSphere<double>(cx,cy,cz,r);
radiusAndCenterFromDualSphere(inputDualSphere, outputRadius, outputCenter);
```

```
std::vector<Mvec<double>> points = extractPairPoint(pp);
Mvec<double> mv4 = surfaceNormal(surface, pointOnSurface);
...
```