Fall 2011 BIOSTAT 615/815 Problem Set #1 - Solutions

Problem 1. Extension of Fisher's Exact Test

```
#include <iostream>
#include <cmath> // for calculating log() and exp()

double logHypergeometricProb(double* logFacs, int a, int b, int c, int d);
void initLogFacs(double* logFacs, int n);

int main(int argc, char** argv) {
  if ( argc != 5 ) {
    std::cerr << "Usage: fullFastFishersExactTest [#row1col1] [#row1col2] [#row2col1] [#row2col2]" << std::endl;
    return -1;
  }

  int a = atoi(argv[1]), b = atoi(argv[2]), c = atoi(argv[3]), d = atoi(argv[4]);
  int n = a + b + c + d;

  double* logFacs = new double[n+1]; // dynamically allocate memory
  initLogFacs(logFacs, n);

  double logpCutoff = logHypergeometricProb(logFacs, a,b,c,d);
  double pTwoSidedFraction = 0;
  double pLessFraction = 0;
  double pGreaterFraction = 0;
  for(int x=0; x <= n; ++x) { // among all possible x
    if ( a+b-x >= 0 && a+c-x >= 0 && d-a+x >=0 ) { // consider valid x
      double l = logHypergeometricProb(logFacs,x,a+b-x,a+c-x,d-a+x);
      double f = exp(l - logpCutoff);
      if ( l <= logpCutoff ) pTwoSidedFraction += f;
      if ( x <= a ) pLessFraction += f;
      if ( x >= a ) pGreaterFraction += f;
    }
  }
  double logpTwoSided = logpCutoff + log(pTwoSidedFraction);
  double logpLess = logpCutoff + log(pLessFraction);
  double logpGreater = logpCutoff + log(pGreaterFraction);
  std::cout << "Two-sided log10(p) = " << logpTwoSided/log(10.) << ", p-value = " << exp(logpTwoSided) << std::endl;
  std::cout << "One-sided (less) log10(p) = " << logpLess/log(10.) << ", p-value = " << exp(logpLess) << std::endl;
  std::cout << "One-sided (greater) log10(p) = " << logpGreater/log(10.) << ", p-value = " <<
       exp(logpGreater) << std::endl;
  return 0;
}

void initLogFacs(double* logFacs, int n) {
  logFacs[0] = 0;
  for(int i=1; i < n+1; ++i) {
    logFacs[i] = logFacs[i-1] + log((double)i);
  }
}

double logHypergeometricProb(double* logFacs, int a, int b, int c, int d) {
  return logFacs[a+b] + logFacs[c+d] + logFacs[a+c] + logFacs[b+d] - logFacs[a] - logFacs[b] - logFacs[c]
         - logFacs[d] - logFacs[a+b+c+d];
}
```

Problem 2 - Pointers and Arrays

Consider the following program `ps-1-2.cpp`.

```
argc = 3            // # of arguments are 3
nv = 3              // nv is copied by value
nr = 3              // nr is bounded to argc by reference
pr[0] = 3           // pr[0] == *pr == argc
pc = ./ps-1-2       // pc == argv[0]
ppc[0] = ./ps-1-2   // ppc[0] == argv[0]
argv[0] = ./ps-1-2 // ppc[0] == argv[0]
c1 = .              // c1 == argv[0][0] = '.'
c2 = l              // c2 == argv[1][2] is 3rd character in 'Hello'
argc = 10           // nr is bound to argc, so argc is also updated
nv = 3              // nv is copied by value, so unaffected by nr
nr = 10             // nr has been update
pr[0] = 10          // pr[0] == argc
pc = Hello          // argv has been advanced, and pc == argv[0]
ppc[0] = Hello      // ppc[0] == argv[0]
c1 = .              // copied by value - does not change
c2 = l              // copied by value - does not change.
```

Problem 3 - Revisiting `towerOfHanoi`

1. 15
2. $2^{n+1} - 1$. $T(n+1) = 2T(n) + 1, T(0) = 1$.
3. print 0 because ra is passed by value, not reference, this not updated.
4.

```
void towerOfHanoi(int n, int s, int i, int d, int& ra) {
  ++ra;
  if ( n > 1 ) towerOfHanoi(n-1,s,d,i,ra);
  std::cout << "Disk " << n << " : " << s << " -> " << d << std::endl;
  if ( n > 1 ) towerOfHanoi(n-1,i,s,d,ra);
}
```