PENN STATE UNIVERSITY
Department of Economics

Econ 597D Sec 001 Computational Economics — Gallant
Homework 7 — Fall 2015
Due Oct 13, 2015

The following program illustrates the use of virtual functions and polymorphism. A copy of it is in src/ch13.

```
#include "libscl.h"
using namespace scl;
using namespace std;

class density_base {
public:
  virtual den_val operator() (REAL) = 0;
  virtual ~density_base() { }
};

class uniform : public density_base {
public:
  den_val operator() (REAL x)
  {
    if (0.0<=x && x<=1.0) return den_val(true,0.0);
    return den_val();
  }
};

class exponential : public density_base {
public:
  den_val operator() (REAL x)
  {
    if (0.0<=x) return den_val(true,-x);
    return den_val();
  }
};

REAL prob(density_base& f, REAL a, REAL b, INTEGER n)
{ //Compute probability with trapezoid rule
  REAL sum = 0.0;
  REAL x = a;
  REAL inc = (b - a)/REAL(n);
  if (f(a).positive) sum += exp(f(a).log_den)/2.0;
  for (INTEGER i=1; i<n; ++i) {
    x += inc;
    if (f(x).positive) sum += exp(f(x).log_den);
  }
  if (f(b).positive) sum += exp(f(b).log_den)/2.0;
  return sum*inc;
}
```

```
int main(int argc, char** argp, char** envp)
{
  REAL a=0.0;
  REAL b=1.0;
  INTEGER g=100;
  uniform u;
  exponential e;
  switch (argc) {
    case 4: g=atoi(argp[3]);
    case 3: a=atof(argp[1]); b=atof(argp[2]); break;
    default: error(string("Usage: ")+argp[0]+" a b [g] "); break;
  }
  cout << prob(u,a,b,g) << '\n';
  cout << prob(e,a,b,g) << '\n';
  return 0;
}
```

Modify class uniform so that it implements the $\mathcal{U}[\alpha, \beta]$ density where the parameters α and β are set by a constructor and the default constructor sets $\alpha = 0$ and $\beta = 1$. Add two more continuous densities that inherit from `density_base` such as the normal $\mathcal{N}(\mu, \sigma^2)$ and the Student $t(\nu)$, where ν is the degrees of freedom parameter. Modify the main so that it computes, using the function `prob` exactly as written, and prints probabilities for the densities instanstantiated at the parameter values $\mathcal{U}[-1, 1]$, $\mathcal{N}(1, 9)$, and $t(6)$ or whatever parameter values are reasonable for the densities you chose to implement.

If you need the gamma function to compute your density, the math libraries on most machines contain the log gamma function `lgamma` and libscl contains `gammln`.

Turn in your code, a sample main that executes it for the values a=-0.5, b=0.5, and g=1000, and the output.