EC 504 – Fall 2022 – Homework Zero

In class put the code on the CCS `/projectnb/alg504/yourname/HW0` Then Yiqin and I will run a script to see if we can look at it. You should put the code in another director such as `/projectnb/alg504/yourname/HW0_working` to practice changing it and put the new version by Sept 13. For fun you can change some of the parameters. No grade but it will show me that you have the system down. In the mean time you should start to read the CLRS Chapters1, 2, and 3 Handouts on GitHub.

1 P^3: Practice Power Program

The exercise is to go to the GitHub and get the prototype code in the file `EC504_2022/HW0_codes`. The code is a main program that calls 3 function to compute the power x^N for large integer N. The first function is the standard C routine. (It actually has to convert the integer power to a floating point. Argh so silly). The next is the slow one. So slow that I stop it before it bores you. The final one the fast multiple squaring routine – but I left out a line for you to complete. *That is the exercise – One line.*

```
double cPower(double x, long int N)
{
  return pow(x, (double)N);
}

double slowPower(double x, long int N)
{
  double pow = 1.0;
  int i;
  for( i = 0;  i < N; i++)
    {
  pow = x*pow;
     }
  // if(i < N)   cout <<"Slow Failed with iteration stop at i = " << i << endl;
  return pow ;
}

double  fastPower(double x, long int N)
{
  double factor = x; // holds x , x^2 , x^4, x^16 etc
  double pow = 1.0;
  while(N > 0)
    {
      if(N%2) pow = factor*pow ; // Update pow
      N = N/2;
      // Update factor by squaring to give the correct increment.
    }
```

```
  return pow;
}
```

There is also a slick bit manipulation code, which C must use for floating point exponents? Easier for integer exponents:

```
double veryfastPower(double x, long int N)
{
    double pow = 1.0;
    for (;;)
    {
      if (N & 1)  //Copies a bit to the result if it exists in both operands.
            pow *= x;
      N >>= 1;  //Binary Right Shift Operator.
      if (!N)  // Check for zero
            break;
        x = x*x;
    }

    return pow;
}
```

The program is compiled automatically by putting `myPower.cpp` in a directory with `makefile` and typing `make -k` on the command line. Then it will run by typing `.\power`

Ok at this point you make a directory `/projectnb/alg504/yourname/HW0` and move it there. Do this right away and we can see if everything is set to go. Then you can fix up the fastPower and see how much faster it is.

Small things to do.

You should read and understand the code as is. Many of this C features will be used in coding exercises.

Also there is one missing code line in

`double fastPower(double x, long int N)`.

I have put the correct output file and a figure in `HW0_codes` to show what the correct output is when you fix it. If you want you can change parameters and graph the data in whatever way you want. I have also give a figure done with `gnuplot` to compare performance scaling. Gnuplot is nice unix utility which we will use later but you may have other graphing programs you like. Anyway this is **no grade practice problem**. **Full credit if we can snarf it from your `HW0` directory in your SCC account.** Don't worry about detail now.

Cheers,

Rich