C++ Exercises

IMPORTANT NOTE: The university cluster machines DO NOT allow programs to be compiled and run from your default network (`P:`) drive. You should work in the `C:\temp` directory and then at the end of each session move your work to your `P:` drive or a memory stick.

Basic Syntax

1. Use Visual Studio to create the "hello world" program as a Console Application. What do you notice when you run the program from within Visual Studio? Can you think of a way to prevent this?

2. Write a program to calculate the (compound) interest on an initial investment of £100 at an annual interest rate of 5% by using a loop structure. **Do not** derive or use the compound interest formula. The idea is to use a loop structure to add the compound interest to the present balance. You should assume that interest is paid annually and print the balance every year for 10 years.

   Modify your code so that the interest rate is entered from the command line. Make sure that your program issues an error if the interest rate is not a real number in the range $0 \leq r \leq 100$.

   Modify your code again so that the interest rate is read from a file called `rate.dat`. Where must the file be located? Try writing a file from the program and then searching for it.

3. Write a program to generate a list of values of x and x^4 for N values of x equally spaced between -10 and 10. You should use a single loop in your program. Try to write the same program using `for`, `while` and `do-while` loops. Which do you prefer?

   Modify your program so that it writes the output into a file. Import the file into Excel and plot your answer for $N = 10$, 100 and 1000.

4. Write a program that stores the first N integers (starting from 1) in a single array. You should allow the value N to be entered from the command line and use dynamic allocation to create the array. Print the array to the screen at the end of your program.

   Modify your program so that the array can be extended by a number M also entered from the command line and then print it again. Sample output is shown below.

```
Enter N: 5
 1 4 9 16 25
Enter M: 3
 1 4 9 16 25 36 49 64
```

C++ Functions

1. Write your own implementation of the function `double power(double x, int y);` that returns the value x^y for integer y. Make sure that your function works correctly for both positive and negative values of y.

2. Write a function to compute the value of $\exp x$ using the first N terms of the power series expansion (N should be an argument to the function):

$$\exp x \approx \sum_{n=0}^{N} \frac{x^n}{n!}$$

   How many terms of the series are required to agree with the built-in exp function? Produce side-by-side output of the two functions for equally-spaced points in the range $x \in [0, 10]$. Import your output into Excel and compare the graphs of the two answers.

3. Write a function

```
double normal_dist(double mu, double sigma, double x);
```

   that returns the probability density function of the normal distribution

$$n(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right)$$

   For the standard normal distribution ($\mu = 0$, $\sigma = 1$), use your function to find the value(s) x_m for which $n(x_m) < 1.0e^{-14}$ to two decimal places?

4. Write a function that calculates the cumulative density function of the standard normal distribution

$$N(x; \mu = 0, \sigma = 1) = \int_{-\infty}^{x} n(x; \mu = 0, \sigma = 1)\ \mathrm{d}x.$$

   You should use your function `normal_dist` as the integrand and take the approximation x_m as the lower limit of integration. Split your domain into M sub-domains and integrate each sub-domain using a standard numerical scheme, say the trapezium rule,

$$\int_a^b f(x)\ \mathrm{d}x = (b-a)\frac{f(a)+f(b)}{2},$$

   or Simpson's rule

$$\int_a^b f(x)\ \mathrm{d}x = \frac{b-a}{6}\Big[f(a) + 4f((a+b)/2) + f(b)\Big].$$

   Compare your results against standard tables or the built-in function in Excel.

Classes

1. Write your own simple `Account` class. You should be able to make deposits and withdrawals and read out the balance — a private double variable. Member functions should be:

   ```
   void Account::withdraw(const double &amount); //Take from account
   void Account::deposit(const double &amount); // Put into account
   double account::balance(void); //Return the balance
   ```

   Make sure that the `Account` constructor function initialises the balance to zero. If you like, add an overloaded constructor function to set an initial balance.

2. Modify your `Account` class so that it writes a record of every transaction into a file. Make sure that you handle file opening and closing properly. If you have multiple accounts where will you write the transactions?

3. Write a derived class `DepositAccount` that inherits from the `Account` class. The account should pay interest at an annual rate that is private member data, but impose a £10 fee for every withdrawal. You should overload the member functions of `Account` where necessary. How will you determine when to pay interest?

4. Write a base class `ContinuousProbabilityDistribution` that contains the member functions:

   ```
   virtual double pdf(const double &x)=0; //Probability density function
   double cdf(const double &x); //Cumulative density function
   double expectation();
   double variance();
   double probability(const double &lower, const double &upper);
   ```

   You should use a numerical integration scheme to calculate the `cdf` (cumulative density function), the `expectation` and the `variance` using the `pdf` function. Do you think that this will be efficient? The probability function should return $P(lower < x < upper)$, which can be calculated by taking the difference of two `cdf` values, or direct numerical integration; which will be more efficient?

   Write three specific derived classes:

   ```
   class NormalDistribution : public ContinuousProbabilityDistribution
   class LogNormalDistribution : public ContinuousProbabilityDistribution
   class GammaDistribution : public ContinuousProbabilityDistribution
   ```

   and overload the `pdf` function in each case:

	Probability density function	Expectation	Variance
Normal distribution	$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right)$	μ	σ^2
Lognormal distribution	$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left[\frac{\log(x/m)}{\sigma}\right]^2\right)$	$m\exp(\frac{1}{2}\sigma^2)$	$m^2 \mathrm{e}^{\sigma^2}\left(\mathrm{e}^{\sigma^2}-1\right)$
Gamma distribution	$f(x) = \begin{cases} \frac{\alpha}{\Gamma(r)}\left(\alpha x\right)^{r-1}\mathrm{e}^{-\alpha x}, & x > 0 \\ 0, & \text{otherwise} \end{cases}$	r/α	r/α^2
	$\Gamma(r) = \int_0^\infty x^{r-1}\mathrm{e}^{-x}\mathrm{d}x, \qquad r > 0$		

Note that μ, σ, m, α and r are parameters that should be member data of the appropriate distribution and the Gamma distribution should have a member function that calculates the Gamma function.

Compare your numerically evaluated expectations and variances with the analytic values presented above. How would you improve the class?

Interfacing C++ and Excel

1. Create your own DLL that implements the simple `square()` function from the notes.

   ```
   double __stdcall square(double &x) {return x*x;}
   ```

   Write an Excel Workbook that uses the square function to calculate the squares of the numbers 1 to 100 and plots your results as a graph.

2. Modify your DLL so that it also includes a function that calculates the balance after N years of compound interest on an initial investment X at an annual interest rate $r(0 \leq r \leq 1)$, assuming that interest is added m times in a year. In other words, the function should return the value of the formula

   $$B = X\left(1 + \frac{r}{m}\right)^{Nm}.$$

   and take the arguments, X, r, m and N.

   Write a Workbook that calls your new function from Excel. Is it faster or slower to use the C++ function, or compute the value in Excel directly?

3. Write a wrapper function to your `PDF` classes so that you can call their member function `probability()` from within Excel. You will either have to write a separate function for each PDF class, or you can write one function that takes an additional (integer) argument that specifies which class to use.

4. Type in the example C++ program from the notes that was used to generate the graph of $f(x) = \mathrm{e}^{-x}\sin(x)$. Check that you can compile and run the program. Modify the program so that it produces a single graph of the three different mathematical functions

   $$f(x) = \sin(x), \quad g(x) = \cos(x), \quad h(x) = \tan(x),$$

   over the range $x \in [0, 2\pi]$.

   Modify the program so that it contains a single C++ function that can be used to plot the graph of any mathematical function over any desired range.