Core questions

Q3.1 Write a function swap that takes two integer parameters and swaps their values, so that the code

```cpp
int a = 5, b = 6;
swap(a, b);
std::cout << a << " , " << b << std::endl;
```

produces the output

6, 5

Since the swap function alters its parameters, think about how those parameters must be passed (by value, by reference, or by const reference?).

(Functions, output parameters)

Q3.2 The following function was written to return the maximum and minimum values in a vector:

```cpp
// given a vector v, returns the maximum and minimum elements in maxVal and minVal
void MaxMin(std::vector<double> v, double maxVal, double minVal)
{
    int s = v.size();
    if (s == 0) return;
    maxVal = minVal = v[0];
    for (int i = 0; i < s; i++)
    {
        if (v[i] < minVal) minVal = v[i];
        if (v[i] > maxVal) maxVal = v[i];
    }
}
```

As written, all three arguments are passed by value. Should each argument be passed by value, by reference, or by const reference? How should the function prototype (the first line) be written?

(argument passing)

Q3.3 The trapezium rule with \( n \) steps approximates an integral of a function \( f \) by

\[
\int_a^b f(x) \, dx \approx h \left( \frac{1}{2} f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2} f(x_n) \right),
\]

where \( h = (b - a)/n \) is the step size and \( x_i = a + h \cdot i \).

Write a function that takes \( a, b \) and \( n \) as parameters, and returns the value of the integral (1), as calculated by the trapezium rule. The integrand \( f(x) \) should be implemented using a second function, with the prototype

```cpp
double f(double x);
```
Use your code to evaluate
\[ I_1 = \int_0^\infty e^{-x^2} \, dx \]

Think about the number of points \( n \) required to obtain an accurate answer, and the infinite limit of the integral can be approximated numerically. Try varying \( n \). How close can you get to the true answer, \( \sqrt{\pi}/2 = 0.886226925452758 \ldots \)?

Suppose now that we have an integrand containing a parameter \( \kappa \geq 0 \),
\[ I_2(\kappa) = \int_0^\infty \sin(\kappa x) e^{-x^2} \, dx, \]
and that we wish to calculate the integral for many values of \( \kappa \). Extend your integrand and trapezium rule functions to incorporate \( \kappa \) (for example, by adding an additional parameter to the functions, or by using a global variable). You may find that this function needs many more points to evaluate accurately than \( I_1 \). Using a numerical search, find the value of \( \kappa \) at which \( I_2(\kappa) \) is maximised (hint — it is between \( \kappa = 1 \) and \( \kappa = 2 \)).

Q3.4 Remind yourself of the syntax for \texttt{std::vector} objects by creating a new vector of 10 \texttt{double}s, assigning values of your choice to each element, and displaying each element.

Write an algorithm to reverse the order of the elements in the \texttt{vector}, and display each element again.

\texttt{(std::vector syntax)}

Q3.5 Write a function that, given a minimum and maximum \( x_{\text{min}} \) and \( x_{\text{max}} \) and a number of elements \( n \), returns returns a \texttt{vector<double>} with \( n \) equally spaced entries from \( x_{\text{min}} \) to \( x_{\text{max}} \):
\[ x_i = x_{\text{min}} + ih, \quad \text{for} \quad i = 0, 1, \ldots, n - 1 \quad h = \frac{x_{\text{max}} - x_{\text{min}}}{n - 1} \]
(You may recognise this as the \texttt{linspace} function from MATLAB).

How might you implement a ‘\texttt{logspace}’ function, which takes the same parameters and also returns a vector of \( n \) elements ranging from \( x_{\text{min}} \) to \( x_{\text{max}} \) but one in which the ratio between each pair of adjacent elements \( x_i \) and \( x_{i+1} \) is constant?

Q3.6 Implement your own version of a function for calculating the power of a number \( a^b \). You should implement two overloaded versions of the function, one for integer \( b \),
\[
\texttt{double power(double a, int b);} \]
that uses repeated multiplication to evaluate the power; and one for real \( b \),
\[
\texttt{double power(double a, double b);} \]
that uses the \texttt{exp} and \texttt{log} functions. Add \texttt{cout} statements to the two functions, to indicate which one is being executed, and verify that
\[
\begin{align*}
\text{• power(1.25, 50) calls the int version} \\
\text{• power(1.25, 50.0) calls the double version}
\end{align*}
\]

Q3.7 Given a vector with components \( x_0, \ldots, x_n \) that is non-decreasing (\( i > j \Rightarrow x_i \geq x_j \)), a vector of values \( y_i = y(x_i) \), and a scalar \( x^* \) (\( x_0 \leq x^* \leq x_n \)), write a function which evaluates \( y(x^*) \), by linearly interpolating the data \( y_i \). Your function should have the prototype
double interp(const std::vector<double> &x,
               const std::vector<double> &y, double x_star);

Test your function on the data

\[ x = (-0.1, 0, 0.25, 0.5, 1) \quad y = (1.2, 1.6, 2.5, 2.1, 1.8) \]

Add some checks to make sure the input parameters to your function are consistent (e.g. elements of \( x \) are increasing, \( y \) has the same number of elements as \( x \), make sure \( x_0 \leq x^* \leq x_n \), etc.)