Question 1 (40%, each of the 3 sub-questions below are worth 15%,15% and 10% respectively).
A definite integral can be computed numerically by the trapezoidal approximation,

$$\int_a^b f(x)dx = \frac{h}{2}(f(x_0) + 2f(x_1) + \ldots + 2f(x_{N-1}) + f(x_N)),$$

where $h = (b - a)/N$ and where $x_n = a + nh$ for $n = 0, 1, \ldots, N$. Write a class called DefInt to compute the trapezoidal approximation for a given function f.

The class should contain the following functionality.

1. Private members to hold the values of the integration limits a, b and a pointer to the function f. The function f should be implemented as a function with one float input argument and return a float.
2. A constructor function such that the integration limits and the pointer to the function f can be initiated at the time of creating an object of the class e.g.

   ```
   DefInt MyInt(a,b,f);
   ```

3. A public function ByTrapezoid() taking N as an argument and returning the trapezoidal approximation to the integral when called by,

   ```
   MyInt.ByTrapezoid(N);
   ```

Question 2 (30%, each of the 3 cases below are worth 10% each)**.**
Using a use case explain how:

(a) Function pointers

(b) Virtual functions, and

(c) Function templates,

are used in C++. Use C++ code to illustrate your answers. **Hint:** You may use the trapezoidal approximation in Question 1 to illustrate your answers.

Question 3 (30%, each of the 2 subquestions below carry equal marks).

(a) Explain (without writing C++ code) how Monte Carlo can be used to compute the Greeks in option pricing. You may explain your answer using an example.

(b) Describe an appropriate variance reduction technique for an Arithmetic Asian option.

Answer 1.

```
class DefInt
{
   private:
      double a,b;
      double (*f)(double x);
   public:
      DefInt(double a_, double b_, double (*f_)(double x))
         {a=a_; b=b_; f=f_;}
      double ByTrapezoid(int N);
      double BySimpson(int N);
};

double DefInt::ByTrapezoid(int N)
{
   double h=(b-a)/N;
   double Result=0.5*f(a);
   for (int n=1; n<N; n++) Result+=f(a+n*h);
   Result+=0.5*f(b);
   return Result*h;
}


double f(double x){return x*x*x-x*x+1;}

int main()
{
   double a=1.0;
   double b=2.0;
   DefInt MyInt(a,b,f);
   int N=10;
   cout << "Trapeziodal approximation = "
        << MyInt.ByTrapezoid(N) << endl;
   return 0;
}
```

Answer 2 (30%, each of the 3 sub-questions below is worth 10%).
If the trapezoidal example is used then the student is expected to explain the issue with function pointers and then illustrate a solution with virtual functions and

function templates.

Answer 3. (a) (Sketch). Expected to provide formula for approximating the derivative of the pricing function and explain how the Monte Carlo paths used for pricing can be reused in some cases.

(b) (Sketch). Suppose we want to compute $\mathbb{E}(X)$ and we already know that some other random variable Y is close to X, and that we already know $y = \mathbb{E}(Y)$. The variable Y is called the control variate and the estimate,

$$\mathbb{E}(X) = \mathbb{E}(X - Y) + y,$$

allows us to reduce the error with out increasing the number of sample paths. A good example of this is in the pricing of an arithmetic Asian option, with a geometric Asian option acting as the control variate.