C++: Practical session 11

1 Simple templates

Write a templated function `sum` that takes a vector of elements of a type and returns their sum. (Do not worry about integer overflow at this stage.) Write specializations of this that perform an accurate sum of `float`s and `double`s. (See Kahan summation).

1.1 Extensions

The code for `float` and `double` in the above should be virtually identical. Create a simple class like the following:

```
template<typename T>
class IsReal{
  static const bool value;
};
```

where `value` is true iff T is a floating-point type and use this to determine whether the simple integral form of the summation is used, or the Kahan version.

For this you should just use specialisations for both `float` and `double`. You will have to specialise the function fully for this.

You could also use `std::is_floating_point<T>::value` which is `true` for `float` and `double` but `false` otherwise.

2 Meta-programming

Enter the following program:

```
#include <iostream>

template<int A, int B>
struct GCD
{
  static const int value = GCD<B, A % B>::value;
};

template<int A>
struct GCD<A,0>
{
  static const int value = A;
};

int main(void)
{
  std::cout << GCD<10,9>::value << std::endl;
  std::cout << GCD<10,4>::value << std::endl;
  std::cout << GCD<55,22>::value << std::endl;
}
```

When is the GCD calculated? When might this sort of operation be useful?

2.1 Extensions

1. Write a template meta-program that allows the compiler to optimize an integer power appropriately, using as few multiplications as possible.

2. Write a template meta-program that determines whether an integer is prime or not.