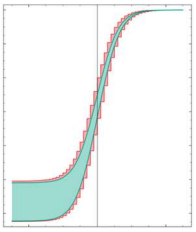

The Interval Arithmetic Project

25% of the Semester's Laboratory Assessment. Your work must be submitted via the Moodle link: Late submission without an ECF will receive the UoN mandated mark penalties.

Please note the distribution of marks for the sub tasks: They are not intended to be proportional to the "difficulty"

Overall Learning Theme of the Exercise:

Strategical Purpose: *To develop a complete object that is expected to be used as a utility component for other programmers in the same way as we might use complex number objects provided by other programmers.*

Focussed objective: *To make professional judgements about the sensible interface, i.e. what functionality should be included, to provide robust implementations and to provide a convincing testing procedure given the time constraints.*

Focussed objective: *To be able to implement operator overloading, in particular for the mathematical operations such as +. To understand the role of polymorphism in this context.*

Focussed objective: *To emphasise that we separate the "What we want to do" of design from the "how we do it" of implementation.*

Reflection:

- Recall our basic "all well designed objects have…" policy
- Decide *what* functionality your class will be providing to users of it and how the function interfaces are specified. Knowing *how* the functionality will be achieved is not part of this *design* exercise.

Interval Arithmetic: In many branches of computational science, a serious issue is the numerical rounding that occurs when calculations are performed with the finite precision representation of floating-point numbers used in CPUs. A way of managing this source of error is the use of *Interval Arithmetic*. With Interval Arithmetic, numerical quantities are specified by the range of values they definitely lie between, i.e. a value v may be characterised by the interval $\{v_{min}, v_{max}\}$ denoting that it is certain that $v_{min} \leq v \leq v_{max}$. As an illustration, π might be expressed as the interval {3.13, 3.15}
By performing a set of calculations using such intervals a measure of the uncertainty of the final result is obtained. This is often used to speed up intensive sequences of operations, for example involving the many pixels in a computer graphics image. (For example, let us say that if the distance of a pixel on a game character from the camera is <1 m then we will draw it in a different way: First we try doing a fast, *rough* calculation of the distance to get $\{d_{min}, d_{max}\}$ and as long as d_{min} >1 m then we can make the decision about the drawing method with certainty. *We didn't need the exact distance value to make the decision, just confidence that it is >1m.*)

The following basic rules define how to arithmetically combine pairs of *intervals,* $\{a_{min}, a_{max}\}$ and $\{b_{min}, b_{max}\}$, to give a result *interval* $\{c_{min}, c_{max}\}$

c=a+b: $\{c_{min}, c_{max}\}=\{a_{min}+ b_{min}, a_{max}+ b_{max}\}$

c=a-b: $\{c_{min}, c_{max}\}=\{a_{min}- b_{max}, a_{max}+ b_{min}\}$

c=a*b: $\{c_{min}, c_{max}\}=\{$ The minimum of $a_{min}*b_{min}, a_{min}*b_{max}, a_{max}*b_{min}, a_{max}*b_{max}$
The maximum of $a_{min}*b_{min}, a_{min}*b_{max}, a_{max}*b_{min}, a_{max}*b_{max}\}$

c=a/b: $\{c_{min}, c_{max}\}=\{$ The minimum of $a_{min}/b_{min}, a_{min}/b_{max}, a_{max}/b_{min}, a_{max}/b_{max}$
The maximum of $a_{min}/b_{min}, a_{min}/b_{max}, a_{max}/b_{min}, a_{max}/b_{max}\}$

Initial Overall Task Specification (Budget: 60 mins work): Your task is to develop a class to encapsulate interval arithmetic. Use of this class for computations should behave as similarly as possible to the use of the intrinsic types, floats or doubles etc. Your class must contain two member variables, *min* and *max*, whose values can only be changed by member functions and friends of the class, but whose values can be read by all. You are specifically required to ensure that your class can be used by the main code given below.

```
int main(){
        interval x;                 // Default initialisation
        interval y(3.0,3.1);        // Initialisation from complete data
        interval z(7);              // Sensible (?) initialisation from a single float
        interval a(x);
        a=x;
        interval *intptr=new interval[4];
        delete[] intptr;
}
```

Note well: The client states that they may come back to you with an *Upgrade* request (see below)

The following sub task counts **30%** of the mark for this exercise

Sub Task Specification (Budget: 30 mins work): Produce a single .h file containing your *design* for the *interface* for the *Interval Arithmetic* class described in the *Initial Overall Specification above*

The following sub task counts **30%** of the mark for this exercise

Sub Task Specification (Budget: 60 mins work): Produce a single .cpp file containing your *implementations* of your *design* of the *Interval Arithmetic* class described in the *Initial Overall Specification above*

The following upgrade task counts **40%** of the mark for this exercise

***Upgrade* Task Specification (Budget: 60 mins work):**

Add to your object, *in the expected manner,* the required functionality necessary to permit the following main program to be used.

```
int main(){

        interval x(3.0,3.1);        // Initialisation from complete data
        interval y(7);              // Sensible (?) initialisation from a single float

        interval a=x+y;
        interval b=x-y;
        interval c=x*y;
        interval d=x/y;

        interval p(x);
        p+=a; // and as a professional you realise that this suggests you should also add?

        cout<<d;
        cin>>a;

        float f(5.);

        interval g=a+f; // and as a professional you realise that this suggests you should also add?

        // A challenge for that final 1/50 mark – i.e. just to prove you can and beat the crowd
        interval h=f+a;
}
```

Project Review: Observe how the first sub task was very much the design phase – the clever bit. We decided what data should be held in the object and what operations were going to be permitted and or made available. We recall that in large projects, we could have circulated just that header file to the rest of the team for their use whilst we undertook the implementation.