

## Struct Questions

This problem involves intervals of real numbers. An interval  $[a,b]$ , where  $a \leq b$ , is the set of all real numbers  $x$  such that  $a \leq x \leq b$ . Two intervals  $[a,b]$  and  $[c,d]$  overlap if they have some point in common.

for example, the interval  $[1.0, 2.0]$  is shown below:



Each of the intervals  $[0.0, 1.5]$ ,  $[1.5, 2.5]$ ,  $[1.25, 1.75]$  and  $[0.0, 3.0]$  overlaps the interval  $[1.0, 2.0]$ .

An interval collection is a collection of nonoverlapping intervals. For example, the interval collection in the diagram below consists of the intervals  $[-3.5, -0.5]$ ,  $[-0.25, 1.25]$ , and  $[1.75, 3.25]$ .



The following definitions of *IntervalType* and *CollType* are used to represent intervals and interval collections, respectively.

```
struct IntervalType
{
    double low, high; // low <= high
};
struct CollType
{
    apvector<IntervalType> intvls(MAX); // MAX is a positive integer constant
    int count; // number of intervals in the collection
};
```

### Part (a)

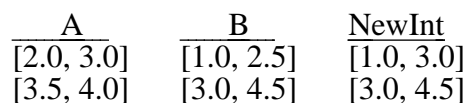
Write the function *Overlap* whose header is given below. *Overlap* returns *true* if its two *IntervalType* parameters represent overlapping intervals and returns *false* otherwise.

```
bool Overlap( IntervalType a, IntervalType b)
```

### Part (b)

Write the function *DoUnion* whose header is given below. *DoUnion* has three parameters, *a*, *b*, and *NewInt*. *a* and *b* represent overlapping intervals. Upon termination of *DoUnion*, *NewInt* represents the union of the intervals represented by *a* and *b*.

For example:



```
void DoUnion(IntervalType a, IntervalType b, IntervalType & NewInt)
```

### Part (c)

Write the function *AddInterval* whose header is given below. *AddInterval* adds interval *AddInt* to the collection of nonoverlapping intervals *Coll* creating the new collection of nonoverlapping intervals *NewColl*. The final value of the parameter *NewColl* is the collection that includes all intervals in *Coll* that do not overlap with *AddInt*, as well as the single interval that is the union of *AddInt* and all intervals in *Coll* that do overlap with *AddInt*.

For example:

*AddInt*  
[0.0, 3.0]

*Coll*  
[2.0, 2.75], [-1.0, 1.0], [4.5, 6.0], [-3.0, -1.5]

*NewColl* (after the call *AddInterval*(*AddInt*, *Coll*, *NewColl*))  
[-3.0, -1.5], [4.5, 6.0], and [-1.0, 3.0]

Note that [0.0, 3.0] overlaps with [2.0, 2.75] and [-1.0, 1.0]; thus, *NewColl* includes [-1.0, 3.0].

Complete function *AddInterval* below the following header. In writing *AddInterval* you may call functions *Overlap* and *DoUnion*. Assume that *Overlap* and *DoUnion* work as specified regardless of what you wrote in parts (a) and (b).

```
void AddInterval(IntervalType AddInt, CollType Coll, CollType & NewColl)
// pre: Coll represents a collection of nonoverlapping intervals
```