

Ejercicio 1 (Topk)

En la clase SecuenciaEnteros

```
void Ordena_por_Insercion(){
    int izda, i;
    int a_desplazar;
    // OJO! Cambiamos a_desplazar < vector_privado[i-1] por
    // a_desplazar > vector_privado[i-1] para ordenarlos de mayor a menor
    for (izda = 1; izda < total_utilizados; izda++){
        a_desplazar = vector_privado[izda];
        for (i = izda; i>0 && a_desplazar > vector_privado[i-1]; i--){
            vector_privado[i] = vector_privado[i-1];
            vector_privado[i] = a_desplazar;
        }
    }
}

SecuenciaEnteros Topk(int k){
    SecuenciaEnteros mayores, secIntermedia;
    for(int i=0; i<total_utilizados; i++){
        secIntermedia.Aniade(Elemento(i));
    }
    if(k > total_utilizados)
        k = total_utilizados;
    secIntermedia.Ordena_por_Insercion();
    for(int i=0; i<k; i++){
        mayores.Aniade(secIntermedia.Elemento(i));
    }
    return mayores;
}
```

Ejercicio 2 (Intervalo)

a)

```
bool SonIguales(double uno, double otro) {
    return fabs(uno-otro) <= 1e-6;
}
```

```
class Intervalo{
private:
    double cotaInferior, cotaSuperior;
    char AbiertoCerrado_Izqd, AbiertoCerrado_Dcha;
public:
    Intervalo(): cotaInferior(1.0), cotaSuperior(0.0),
        AbiertoCerrado_Izqd('[', AbiertoCerrado_Dcha(']')
    {
    }

    Intervalo(double topeInf, double topeSup, char izqd, char dcha){
        cotaInferior = topeInf;
        cotaSuperior = topeSup;
        AbiertoCerrado_Izqd = izqd;
        AbiertoCerrado_Dcha = dcha;
    }
    .....
    .....
};
```

b)

```
bool Vacio(){
    bool vacio;
    if(cotaInferior > cotaSuperior)
        vacio = true;
    else if(SonIguales(cotaInferior, cotaSuperior) && (AbiertoCerrado_Izqd == '(' && AbiertoCerrado_Dcha == ')') )
        vacio = true;
    else if (cotaInferior > cotaSuperior)
        vacio = true;
    else
        vacio = false;
    return vacio;
}
```

c)

En la clase SecuenciaDouble

```
void Aniade(double nuevo);
double Elemento(int indice);
int TotalUtilizados();
```

En la clase Intervalo

```
bool Esta_Dentro(double x){
    bool esta;
    if(x < cotaInferior || x > cotaSuperior)
        esta = false;
    else if(SonIguales(x, cotaInferior) && (AbiertoCerrado_Izqd == '('))
        esta = false;
    else if(SonIguales(x, cotaSuperior) && (AbiertoCerrado_Dcha == '('))
        esta = false;
    else
        esta = true;
    return esta;
}
```

```
SecuenciaDouble Contenidos(SecuenciaDouble sec){
    SecuenciaDouble secDentro;
    for(int i= 0; i< sec.TotalUtilizados(); i++)
        if(Esta_Dentro(sec.Elemento(i)))
            secDentro.Aniade(sec.Elemento(i));
    return secDentro;
}
```

```
};
```

Ejercicio 3 (Tabla)

a)

```
class TablaRectangularReales{
    private:
        static const int MAX_FIL = 50;
        static const int MAX_COL = 40;
        double matriz_privada[MAX_FIL][MAX_COL];
        int util_fil;
        const int util_col;
```

b)

En la clase SecuenciaDouble

```
double Media(int izq, int dcha){
    double media, suma=0.0;
    for(int i = izq; i < dcha+1; i++)
        suma += vector_privado[i];
    media = suma/(dcha-izq +1);
    return media;
}
```

En la clase TablaRectangularReales

```
TablaRectangularReales PromedioRelativo(){
    TablaRectangularReales otra(util_col);
    SecuenciaDouble tmp, nueva_fila;
    for(int i=0; i < util_fil; i++){
        tmp = Fila(i);
        for(int j =0; j < util_col; j++){
            if(i>j)
                nueva_fila.Aniade(otra.Elemento(j, i));
            else if(i==j)
                nueva_fila.Aniade(Elemento(i, j));
            else if(i<j)
                nueva_fila.Aniade(tmp.Media(j, util_col-1));
        }
        otra.Aniade(nueva_fila);
        nueva_fila.EliminaTodos();
    }
    return otra;
}
```

c)

En el main()

```
TablaRectangularReales original, resultado;
.....
.....

resultado = original.PromedioRelativo();
.....
```