```
/* Examen Junio 2015*/
//Ejercicio 1.-
class celda{
    private:
        double valor
        Celda * siguiente;
    public:
        inline Celda(){valor=0;celda=0}
        inline Celda(double valor){this->valor=valor;siguiente=0;}
        inline double obtenerValor(return valor);
        inline asignarValor(this->valor=valor);

}

class Lista{
    private:
        Celda * primerValor;
    public:
        inline Lista(){primerValor=0;}
        inline Lista(double valor){primerValor=new Celda(valor);}
        void insertarInicio(double valor);
        void imprimir(const Lista & lista);
        void imprimirUltimoAPrimero(const Lista & lista);
}
void insertarInicio(double valor){
    if(primerValor==0)
        primerValor=new Celda(valor);
    else{
        Celda *ptr=new Celda(valor);
        Celda *aux=primerValor;
        primerValor=prt;
        primerValor->siguiente=aux;
    }

}

void imprimir(const Lista & lista){
    Celda *ptr=lista.primerValor;
    while(ptr->siguiente!=0){
        cout << ptr->obtenerValor(); << " ";
    }
    cout << endl;

}

void imprimirUltimoAPrimero(const Lista & lista){
    Lista auxiliar;
    Celda *ptr=lista.primerValor;
    while(ptr!=0){
        auxiliar.insertarInicio(ptr->obtenerValor());
        ptr=ptr->siguiente();
    }
    imprimir(auxiliar);
}

//Ejercicio 2.-
class Pareja{
    int dato;
    int nveces;
}

class Frecuencias{
    private:
        Pareja * parejas;
        int nparejas;
        void liberarEspacio();
        void reservarMemoria(int nparejas);
        void Frecuencias::copiarDatos(const Frecuencias & otro);

    public:
        Frecuencias();
        Frecuencias(const Frecuencias & otro);
        ~Frecuencias();
```

```
        Frecuencias & operator=(const Frecuencias & otro);
        ostream & operator<<(ostream & flujo,const Frecuencias & objeto);


    void Frecuencias::liberarEspacio(){
        if(parejas!=0){
            delete [] parejas;
            nparejas=0;
        }
    }
    void Frecuencias::copiarDatos(const Frecuencias & otro){
        for(int i=0;i<nparejas;i++){
            this->parejas[i]=otro.parejas[i];
        }
    }
    void Frecuencias::reservarMemoria(int nparejas){
        parejas = new Parejas[nparejas];
        this->nparejas=nparejas;
    }
    Frecuencias::Frecuencias(){
        nparejas=0;
        parejas=0;
    }
    Frecuencias::~Frecuencias(){
        liberarEspacio();
    }

    Frecuencias::Frecuencias(const Frecuencias & otro){
        reservarMemoria(otro.nparejas);
        copiarDatos(otro);
    }


    Frecuencias & Frecuencias::operator=(const Frecuencias & otro){
        if(this!=otro){
            liberarEspacio();
            reservarMemoria(otro.nparejas);
            copiarDatos(otro);
        }
        return *this;
    }

    ostream & operator<<(ostream & flujo,const Frecuencias & objeto){
        for(int i=0;i<objeto.nparejas)
            flujo<< objeto.parejas[i].dato << " "<< objeto.parejas[i].nveces << " ";
        flujo << endl;
        return flujo;
    }

}
```