Ejercicios ED

Ejercicio 24: sign_split

```
void sign_split(list<int> &L, vector<list<int>> &VL ){

    bool signo = *(L.begin()) >=0 ? 0:1;
    list<int> l1;
    for (list<int>::iterator it = L.begin(); it!= L.end(); it++){
        if (signo && *it>=0 || (!signo && *it<0)){
            VL.push_back(l1);
            signo =0;
            l1.clear();
        }
        else{
            l1.push_back(*it);
        }
    }

}
```

Ejercicio 25: map

```
class map{

    private:
        vector<pair<int,string>> r;
        iterator lower_bound(int x);
    public:
        iterator find(int x);
        pair<bool, iterator> insert(int x);
        string retrieve(int x);

    //Me da pereza implementar todo pero es como siempre, hago lo
difisil
        class iterator{

            private:
                vector<pair<int, string>> it;
            public:

                iterator();
                iterator(const iterator);
                iterator & operator =(const iterator it);
                pair<int, string>& operator* ();
                bool operator ==(const iterator& it);
                bool operator!=(const iterator& it);
                iterator& operator++();
        };
```

```
        iterator& begin();
        iterator& end();


    };

//Devuelve la primera aparición del elemento en el mapa
map::map::iterator& lower_bound(int x){
    bool encontrado= false;
    map::iterator ot;
    for (map::iterator it=begin(); it!=end() && !encontrado; it++){
        if (*it.first=x){
            encontrado = true;
            ot = it;
        }
    }
    return it;
}
```

Ejercicio 26: max_sublist

```
void max_sublist(list<int> &L, list<int>& subL){

    list<int> max;
    list<int>sub1;
    for (list<int>::iterator it=L.begin(); it!=L.end(); it++){
        if (*it%2 == 0){
            sub1.push_back(*it);
        }
        else{
            if (sub1.size()>max.size())
                max=sub1;
            sub1.clear();
        }
    }
}
```

Ejercicio 28: has_sum()

```
bool has_sum(set<int>&s, int M){
    if (s.count(M) > 0)
        return true;
    else{
        bool todosmayores=true;
```

```
        for (set<int>::iterator it=s.begin(); it!= s.end() &&
todosmayores; it++){
            if (*it <M)
                todosmayores=false;
        }
        if (todosmayores)
            return false;
        else{

            //Cojo un elemento <M
            bool tiene = false;
        for (set<int>::iterator it=s.begin(); it!= s.end() && !tiene ;
it++){
            if (*it <M){
                copia = s;
                copia.erase(it);
                tiene = has_sum(copia, M-*it);
            }

        }




          return tiene;
        }

    }
}
```

Ejercicio 29: greatest_subset

```
int apariciones(set<char> v, string s){
    int contador =0;
    for (int i=0; i<s.size(); i++){
        if (v.count(s[i])>0)
            contador++;
    }
    return contador;
}

int greatest_subset( vector<set<char>>& vs, string s){
    int posmax=0, posicion=0, apmax=0;
    for (vector<set<char>>::iterator it= vs.begin(); it!
= vs.end(); it++){
        if (apariciones(*it, s)> 0){
            apmax=apariciones(*it,s);
            posmax=posicion;
        }
        posicion ++;

    }
    return posmax;
```

```
}
```

Ejercicio 30: only1

```
set<int> union(set<int> uno, set<int> dos){
    set<int> nuevo=uno;
    for (set<int>::iterator it= dos.begin(); it!= dos.end(); it++){
            nuevo.insert(*it);
    }
    return nuevo;
}

void only1(vector<set<int>>&vs, set<int>&s1){
    set<int> uni;
    for (vector<set<int>>::iterator it=vs.begin(); it!= vs.end(); it++){
        uni=union(uni, *it);
    }
    set<int>::iterator ot;
    for (set<int>::iterator it=uni.begin(); it!= uni.end(); it++){
        if (uni.count(*it)>1){
            while (uni.count(*it)!= 0){
                ot=uni.find(*it);
                uni.erase(ot);
            }
        }
    }
    return uni;

}
```

Ejercicio 31:

hasta el coño

```
int cantidadpares(list<int>& l){
    int contador=0;
    for (list<int>::iterator it=l.begin(); it!=l.end(); it++){
        if (*it%2 ==0)
            contador++;
    }
    return contador;
}

void large_even_list(vector<list<int>>& vl, list<int>&l){
    list<int> hola;
    int contmax=0;
    for (vector<list<int>>::iterator it = vl.begin(); it!= vl.end();
it++){
        if (cantidadpares(*it)>contmax){
            contmax=cantidadpares(*it);
            hola = *it;
        }

    }
```

```
    l=hola;
}
```

Ejercicio 32: interlaced_split

```
void interlaced_split(list<int>&l, int m, vector<list<int>>&vl){
    int pos=0, actual=0;
    //Inicializo el vector
    list<int> hola;
    for (int i =0; i<m; i++)
        vl.push_back(hola);
    for (list<int>::iterator it=l.begin();it!=l.end(); it++, pos++){

        //Se puede hacer con el operador[], pero pa practicar
        actual=pos/m;
        vector<list<int>>::iterator ot=vl.begin();
        for (int i=0; i <actual; i++)
            ot++;
        (*ot).push_back(*it);

    }
}
```

Ejercicio 33: inall

```
bool incluido(set<int>uno, set<int> otro){
    bool loesta=true;
    for (set<int>::iterator ot=uno.begin(); ot!= uno.end(); ot++){
        if (otro.count(*it) ==0)
            loesta=false;
    }
}

bool inall(list<set<int>>&Ls, set<int> &s){
    bool loesta=true;
    bool encontrado=false;
    for (list<set<int>>::iterator it=ls.begin(); it!=ls.end() &&
!encontrado; it++){
        loesta=true;
        for (list<set<int>>::iterator ot = ls.begin(); ot!=ls.end() &&
loesta; ot++)
        {
            if (!incluido(*it, *ot))
                loesta=false;
        }
        if (loesta){
            encontrado=true;
            s=*it;
```

```
        }
    }
}
```