Pro2: indicacions entrega 1

1. Exercici X18215 (Reducció de programes)

En aquest exercici, hem d'aplicar regles sobre l'string d'entrada fins que ja no s'en pugui aplicar cap més. Les regles sempre s'han d'aplicar el més a l'esquerra possible.

Podem tenir un bucle general que crida a una funció `applyOneRule` que mira d'aplicar una sola regla.

1.1. Versió lenta

Una possible manera d'obtenir una versió lenta és que `applyOneRule` recorri totes les posicions de l'string, d'esquerra a dreta, i miri d'aplicar una regla en aquella posició.

Aquesta seria una possible estructura del programa:

```
#include <iostream>
#include <string>

using namespace std;

bool applyOneRule(string &s, int i)
{
    if (s[i] == 'v' and (i+1 == int(s.size()) or s[i+1] != '=')) {
        s[i] = 'E';
        return true;
    } else if ...
    ...
    } else if (i+2 < int(s.size()) and
               s[i] == '(' and s[i+1] == 'E' and s[i+2] == ')' and
               (i == 0 or (s[i-1] != 'i' and s[i-1] != 'w'))) {
      s = s.substr(0, i) + string(1, 'E') + s.substr(i+3);
      return true;
    } else if ...
    ...
    }
    return false;
}
```

```
bool applyOneRule(string &s)
{
    for (int i = 0; i < int(s.size()); i++)
        if (applyOneRule(s, i))
            return true;
    return false;
}

int main()
{
    string s;
    while (cin >> s) {
        while (applyOneRule(s)) ;
        cout << s << endl;
    }
}
```

1.2. Versió ràpida

Per a obtenir una versió ràpida, podem afegir un `vector<char>v` que vagi mantenint la part ja tractada de `s` i a la qual ja se li han aplicat totes les regles possibles, de manera que sabem que no cal tornar a intentar aplicar regles sobre aquestes posicions, i així ens estalviem temps.

A cada pas, afegim al final de `v` un nou caràcter de l'entrada. La funció `applyOneRule` intenta aplicar una regla sobre `v`. Però només cal intentar aplicar-la al final de `v` perquè ja sabem que tota la resta de `v` està en forma normal.

Podem passar també un paràmetre `follow` amb el caràcter que ve després, per tal de comprovar les restriccions sobre quins caràcters no poden seguir a la part esquerra d'una regla.

Aquesta seria una possible estructura del programa:

```
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>

using namespace std;

bool applyOneRule(vector<char> &v, char follow)
{
    int n = v.size();
    if (v[n-1] == 'v' and (follow != '=')) {
        v[n-1] = 'E';
        return true;
    } else if ...
    ...
```

```
    } else if (n >= 3 and
               v[n-3] == '(' and v[n-2] == 'E' and v[n-1] == ')' and
               (n == 3 or (v[n-4] != 'i' and v[n-4] != 'w'))) {
        v.pop_back();
        v.pop_back();
        v.pop_back();
        v.push_back('E');
        return true;
    } else if ...
    ...
    }
    return false;
}

int main()
{
    string s;
    while (cin >> s) {
        vector<char> v;
        for (int i = 0; i < int(s.size()); i++) {
            v.push_back(s[i]);
            char follow = ' ';
            if (i+1 < int(s.size()))
                follow = s[i+1];
            while applyOneRule(v, follow) ;
        }
        string sol;
        for (int i = 0; i < int(v.size()); i++)
            sol += v[i];
        cout << sol << endl;
    }
}
```

2. Exercici X12746 (Vectors amb resize, i push i pop a front i back)

2.1. Versió lenta

Una possible versió lenta d'aquest exercici consistirà simplement en utilitzar un `vector` o un `list` per a simular les comandes.

2.2. Versió ràpida

Els jocs de proves privats poden fer `resize` amb mides molt grans i, a més a més, necessitarem trobar una manera de tenir cost global $n \log n$.

Utilitzar `map` sembla una bona elecció, doncs totes les operacions amb `map` tenen cost com a molt logarítmic.

Podem tenir un `map` índex↦valor, que guardi només els valors dels índexos que han estat explícitament definits per les comandes.

Una de les dificultats és el tractament de `push_front` i `pop_front`, que, a priori, ens obligarien a desplaçar tots els índexos. Una alternativa és mantenir un enter `first` que guardi quin és el primer índex vàlid. Al principi, `first` valdrà 0. Cada comanda `push_front` provocarà un `first--`, i cada comanda `pop_front` provocarà un `first++`, entre altres coses.