Scrivere una funzione **verifica** che prenda in input una collezione S di stringhe, e restituisca true se è possibile suddividere S in due insiemi disgiunti S_1 ed S_2 tale che la somma dei valori delle stringhe nel primo insieme sia uguale alla somma dei valori delle stringhe nel secondo insieme; in caso contrario, la funzione restituisce false. Il valore di una stringa s è un numero intero (positivo o negativo) dato da una funzione val(s), la quale si può supporre essere globale e già implementata.

Si può assumere che:

- S è rappresentato da un vector<string>
- S contiene almeno due stringhe e può contenere più volte la stessa stringa,
- val è una funzione con visibilità globale, prende in input una stringa presente in S, e restituisce il suo valore (un numero intero).

Esempio: la funzione dovrà restituire true poiché è possibile suddividere S in due insiemi tale che la somma dei valori delle stringhe (dati dalla funzione val) sia uguale tra i due insiemi. In questo caso, una possibile suddivisione è la seguente: S_1 = {ab, q, 8a8b} e S_2 = {cde, xyz, q42, ccc}, dove la somma per S_1 è pari a 53, ed è uguale alla somma per S_2.	S = {ab, cde, xyz, q, q42, 8a8b, ccc} val(ab) = 11 val(cde) = 11 val(xyz) = 12 val(q) = 8 val(q42) = 7 val(8a8b) = 34 val(ccc) = 23

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int val(string s){return s.length();}//solo per crearne una al volo

struct Tsoluzione{
     vector<string> S;
     vector<bool> sol;
     int sommaTot;
     int sommaSol;

     Tsoluzione(const vector<string>& _S):S(_S),sommaSol(0){
          sommaTot=0;
          for (auto s:S) sommaTot+=val(s);
     }
};

void stampaSol(const Tsoluzione& bkt)
{
    cout<<"Insieme 1"<<endl;
    for (int i=0; i<bkt.sol.size();i++)
        if (bkt.sol[i]) cout<<bkt.S[i]<<" "<<val(bkt.S[i])<<endl;

    cout<<"Insieme 2"<<endl;
    for (int i=0; i<bkt.sol.size();i++)
        if (!bkt.sol[i]) cout<<bkt.S[i]<<" "<<val(bkt.S[i])<<endl;

    cout<<bkt.sommaTot<<" "<<bkt.sommaSol<<endl;
}

void add(int x, Tsoluzione& soluzione){
  soluzione.sol.push_back(x);
  if (x==1)
     soluzione.sommaSol+=val(soluzione.S[soluzione.sol.size()-1]);

}

void remove(Tsoluzione& soluzione){
    int v=soluzione.sol[soluzione.sol.size()-1];
    soluzione.sol.pop_back();
    int x=soluzione.sol.size();
    if (v==1) soluzione.sommaSol-=val(soluzione.S[x]);

}

bool isComplete(const Tsoluzione & soluzione){
    return (soluzione.sol.size()==soluzione.S.size() &&
soluzione.sommaSol==(soluzione.sommaTot- soluzione.sommaSol);
}
```

```
bool canAdd(int x, const Tsoluzione & soluzione)
{
    if (soluzione.sol.size()>=soluzione.S.size()) return false;
    if (x==0) return true;
    if
((soluzione.sommaSol+val(soluzione.S[soluzione.sol.size()]))>(soluzione.
sommaTot/2)) return false;
    return true;
}

bool solve(Tsoluzione& soluzione) {
    int x = 0;
    while (x < 2) {
        if (canAdd(x, soluzione)) {
            add(x, soluzione);

            if (isComplete(soluzione))
                return true;
            else if (solve(soluzione))
                return true;
            remove(soluzione);
            x++;
        } else
            x++;
    }
    return false;
}

int main(){
    vector<string> S={"a","aa","aaa","aaaa"};
    Tsoluzione bkt(S);
    bool result;
    solve(bkt)? result=true:result=false;

    if (result==false) cout<<"NO";
    else {
        cout<<"YES"<<endl;
        stampaSol(bkt);
    }

}
```