Uno `SmartContract` è un servizio che consente agli utenti di effettuare transazioni che verranno mantenute in una lista d'attesa e successivamente **confermate** e poi **spostate** dentro una lista di transazioni effettivamente completate e non più modificabili detta "blockchain". Una transazione è **confermata** quando la transazione monetaria è stata effettuata. Le transazioni sono mantenute in ordine dalla prima all'ultima inserita. Una transazione mantiene un codice identificativo di al più 10 caratteri, il nome del sorgente avente al più 20 caratteri, il nome del destinatario avente al più 20 caratteri, la somma di denaro che deve essere trasferita dal sorgente al destinatario e un campo che indica se la transazione è confermata oppure no. Implementare le seguenti operazioni che possono essere effettuate su uno `SmartContract`:

- `Inizializza(s)`;
Funzione che inizializza uno smart contract `s`. Inizialmente non sono presenti transazioni né nella blockchain, né nelle transazioni in attesa.
- `AggiungiInAttesa(s, id, src, dst, q)`;
Funzione che inserisce una nuova transazione nelle transazioni in attesa di conferma dello smart contract `s`, dove `id` è il codice della transazione, `src` è il nome del sorgente, `dst` è il nome del destinatario, `q` è la quantità di denaro da trasferire. La transazione **deve ancora essere confermata**. Non ci possono essere due transazioni con lo stesso codice identificativo nella lista di attesa.
- `ConfermaPagamento(s, id)`;
Funzione che modifica l'esito della transazione con codice `id` dello smart contract `s`, in modo che risulti **confermata**. Se non esiste la transazione con codice `id` nella lista di attesa oppure è già confermata, l'operazione restituisce `false` e lascia lo smart contract `s` inalterato; altrimenti restituisce `true`.
- `SpostaInBlockchain(s)`;
Funzione che sposta la transazione confermata inserita meno di recente nella lista d'attesa dello smart contract `s`, mettendola dentro la blockchain. La transazione **non sarà più presente nella lista d'attesa ma solo dentro la blockchain**. Se nessuna transazione confermata è presente nella lista d'attesa la funzione restituisce `false`, altrimenti restituisce `true`.
- `SoldiSpesi(s, src)`;
Funzione che restituisce la quantità di soldi spesi dall'utente con nome `src` nello smart contract `s`. Contano come soldi spesi solo quelli relativi a transazioni presenti nella blockchain.
- `ModificaPagamento(s, src)`;
Funzione che raddoppia l'importo di tutte le transazioni del sorgente `src`, che sono nella lista di attesa e che non sono ancora state confermate. Restituisce il numero delle transazioni modificate.
- `Stampa(s)`;
Funzione di uscita di uno `SmartContract` `s`. La funzione stampa il numero di transazioni nella lista d'attesa racchiuso fra parentesi quadrate, seguito dall'identificatore `id` e dall'importo `q` delle transazioni nella blockchain, racchiusi all'interno di parentesi angolate e separati da '-'. Le transazioni sono stampate nell'ordine da quella inserita meno di recente a quella inserita più di recente nella blockchain.
L'uscita seguente mostra uno `SmartContract` `s` con 5 transazioni nella lista d'attesa. Le transazioni con `id` T1 e T5 sono nella blockchain; l'importo `q` di T1 è pari a 11.0; l'importo `q` di T5 è pari a 13.5. T1 è la transazione inserita per prima nella blockchain; T5 è la transazione inserita per ultima nella blockchain.

[5] <T1-11.0> <T5-13.5>

Mediante il linguaggio C++, realizzare il tipo `SmartContract` definito dalle precedenti specifiche utilizzando le strutture (non utilizzare il tipo classe e non utilizzare il tipo string definito nelle librerie per le stringhe). Gestire le eventuali situazioni di errore.

Esempio di funzione main()

Contenuto dello stream d'output standard mostrato nel commento */** */*

```
int main(){

    SmartContract s;
    Inizializza(s);
    Stampa(s);
    // [0]

    AggiungiInAttesa(s, "T0", "Alessio", "Anna", 25.7);
    AggiungiInAttesa(s, "T1", "Anna", "Alessio", 12.5);
    AggiungiInAttesa(s, "T2", "Alessio", "Anna", 32.1);
    Stampa(s);
    // [3]

    ConfermaPagamento(s, "T1");
    Stampa(s);
    // [3]

    SpostaInBlockchain(s);
    ConfermaPagamento(s, "T0");
    SpostaInBlockchain(s);
    Stampa(s);
    // [1] <T1-12.5> <T0-25.7>

    double quanti = SoldiSpesi(s, "Alessio");
    cout << quanti << endl;
    // 25.7

    ModificaPagamento(s, "Alessio");
    ConfermaPagamento(s, "T2");
    SpostaInBlockchain(s);
    Stampa(s);
    // [0] <T1-12.5> <T0-25.7> <T2-64.2>

    return 0;
}
```