Un `Tragitto` è formato da postazioni numerate a partire da 1. Nel Tragitto transitano persone individuate dal nome. Ciascuna postazione può essere libera o occupata da una persona. Non è possibile inserire persone con lo stesso nome nel `Tragitto`. Si assuma che il nome della persona sia al massimo di 19 caratteri. Il `Tragitto` deve essere progettato sin dall'inizio in maniera tale da poter contenere **un numero potenzialmente illimitato di postazioni** (vedasi operazione +=).

Implementare le seguenti operazioni che possono essere effettuate su un `Tragitto`.

--- PRIMA PARTE ---

✓ `Tragitto t;`
Costruttore di default, che inizializza un `Tragitto t`. Tale `Tragitto` contiene due postazioni, entrambe inizialmente libere.

✓ `t.inserisci(s);`
Operazione che inserisce nella prima postazione del `Tragitto` una persona di nome `s`. L'operazione fallisce se la prima postazione è occupata, oppure se nel `Tragitto` esiste già una persona di nome `s`. L'inserimento fallisce infine nel caso in cui la stringa s sia di lunghezza superiore a quella consentita.

✓ `t.avanza(j);`
Operazione che avanza di una posizione la persona che occupa la postazione `j`. Se la postazione `j+1` è occupata, l'operazione lascia il `Tragitto` inalterato. Se la postazione `j` è l'ultima del `Tragitto`, la persona esce dal `Tragitto`.

✓ `cout << t;`
Operatore di uscita per il tipo Tragitto. L'uscita ha la forma seguente:

```
[1] /////
[2] Rossi
[3] /////
[4] Verdi
[5] /////
```

In questo esempio, il `Tragitto` contiene 5 postazioni. La prima, la terza e la quinta sono libere. La seconda è occupata da una persona di nome Rossi, la quarta da una persona di nome Verdi. Tra la parentesi quadra chiusa e l'inizio del nome (o dei 5 caratteri '/') **c'è un solo spazio**.

--- SECONDA PARTE ---

✓ `t += n;`
Operatore di somma e assegnamento, che aggiunge `n` postazioni libere al `Tragitto t`. Le postazioni vengono inserite in fondo al `Tragitto`.

✓ `Tragitto t1(t);`
Costruttore di copia, che inizializza un `Tragitto t1` con il valore del `Tragitto t`.

✓ `~Tragitto();`
Distruttore.

Mediante il linguaggio C++, realizzare il tipo di dato astratto definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc.

```cpp
// file main.cpp
#include<iostream>
#include "compito.h"
using namespace std;

int main(){
    cout<<"--- PRIMA PARTE ---"<<endl;
    cout<<"Test del costruttore"<<endl;
    Tragitto t;
    cout << t << endl;

    cout<<"Test della inserisci"<<endl;
    t.inserisci("Verdi");
    cout<<t<<endl;

    t.inserisci("Rossi");    // deve fallire, perchè la prima postazione è occupata
    cout<<t<<endl;

    cout<<"Test della avanza"<<endl;
    t.avanza(1);
    cout<<t<<endl;

    cout<<"Altro test della inserisci"<<endl;
    t.inserisci("Rossi");
    cout<<t<<endl;

    cout<<"--- SECONDA PARTE ---"<<endl;
    cout<<"Test dell'operatore +="<<endl;
    t += 3;
    cout<<t<<endl;

    cout<<"Altri test della avanza"<<endl;
    t.avanza(2);
    t.avanza(3);
    cout<<t<<endl;
    t.avanza(1);
    cout<<t<<endl;

    cout<<"Provo ad inserire di nuovo Verdi (l'inserimento deve fallire')"<<endl;
    t.inserisci("Verdi");  // deve fallire perchè un Verdi esiste già
    cout<<t<<endl;

    {
        cout<<"Test del costruttore di copia"<<endl;
        Tragitto t1(t);
        cout<<t1<<endl;
    }
    cout<<"Test del distruttore (e' appena stato distrutto l'oggetto t1)"<<endl;
}
```

USCITA ATTESA

```
--- PRIMA PARTE ---
Test del costruttore
[1] /////
[2] /////

Test della inserisci
[1] Verdi
[2] /////

[1] Verdi
[2] /////

Test della avanza
[1] /////
[2] Verdi

Altro test della inserisci
[1] Rossi
[2] Verdi

--- SECONDA PARTE ---
Test dell'operatore +=
[1] Rossi
[2] Verdi
[3] /////
[4] /////
[5] /////

Altri test della avanza
[1] Rossi
[2] /////
[3] /////
[4] Verdi
[5] /////

[1] /////
[2] Rossi
[3] /////
[4] Verdi
[5] /////

Provo ad inserire di nuovo Verdi (l'inserimento
deve fallire')
[1] /////
[2] Rossi
[3] /////
[4] Verdi
[5] /////

Test del costruttore di copia
[1] /////
[2] Rossi
[3] /////
[4] Verdi
[5] /////
```