Il tipo di dato astratto `ProntoSoccorso` memorizza i pazienti in attesa di un pronto soccorso. I pazienti sono identificati da una stringa contenente il proprio nome e cognome. La stringa ha almeno un carattere e ne può contenere al massimo 20. A ogni paziente in arrivo viene associato un livello di priorità codificato con quattro colori, in ordine di priorità crescente: bianco, verde, giallo e rosso.
I pazienti vengono trattati in ordine di priorità. A parità di livello di priorità, i pazienti vengono trattati secondo l'ordine di arrivo. Dopo che un paziente è stato trattato, lascia il pronto soccorso.
Implementare le seguenti operazioni che possono essere effettuate su un `ProntoSoccorso`:

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

- ✓ **`ProntoSoccorso ps;`**

Costruttore che inizializza un `ProntoSoccorso`. Inizialmente, non ci sono pazienti in attesa.

- ✓ **`ps.ricovero(nome, liv);`**

Ricovera il paziente di nome `nome`, assegnandogli il livello di priorità `liv`. La funzione non modifica lo stato di `ps` se `nome` assume un valore non valido.

- ✓ **`ps.prossimo(nome);`**

Seleziona il prossimo paziente per il trattamento secondo l'ordine di priorità. Il nome del paziente selezionato viene restituito nella stringa `nome`. Il valore di ritorno della funzione è 0 se non ci sono pazienti in attesa, altrimenti è 1.

- ✓ **`cout << ps;`**

Operatore di uscita per il tipo `ProntoSoccorso`. L'uscita ha il seguente formato:

```
Numero pazienti: 4
[CODICE ROSSO]
->Mario Rossi
->Giuseppe Gialli
[CODICE GIALLO]
[CODICE VERDE]
->Paolo Verdi
[CODICE BIANCO]
->Maria Neri
```

L'output mostrato corrisponde a un `ProntoSoccorso` avente quattro pazienti in attesa: due sono in codice rosso, uno in codice verde e uno in codice bianco.

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

- ✓ **`ProntoSoccorso ps1(ps);`**

Costruttore di copia per il tipo `ProntoSoccorso`, che crea un pronto soccorso `ps1` uguale a `ps`.

- ✓ **`ps1 = ps;`**

Operatore di assegnamento per il tipo `ProntoSoccorso`, che rende `ps1` uguale a `ps`.

- ✓ **`~ProntoSoccorso();`**

Distruttore.

Mediante il linguaggio C++, realizzare il tipo di dato astratto **`ProntoSoccorso`**, definito dalle precedenti specifiche. **Gestire le eventuali situazioni di errore**.

continua sul retro del foglio

OUTPUT ATTESO DAL PROGRAMMA

```
---PRIMA PARTE---
Test del costruttore di default:
Numero pazienti: 0
[CODICE ROSSO]
[CODICE GIALLO]
[CODICE VERDE]
[CODICE BIANCO]

Test di ricovero:
Numero pazienti: 4
[CODICE ROSSO]
->Mario Rossi
->Giuseppe Gialli
[CODICE GIALLO]
[CODICE VERDE]
->Paolo Verdi
[CODICE BIANCO]
->Maria Neri

Test di prossimo:
Prossimo: Mario Rossi
Prossimo: Giuseppe Gialli
Prossimo: Paolo Verdi

Numero pazienti: 1
[CODICE ROSSO]
[CODICE GIALLO]
[CODICE VERDE]
[CODICE BIANCO]
->Maria Neri

---SECONDA PARTE---
Test dell'operatore =:
Numero pazienti: 1
[CODICE ROSSO]
[CODICE GIALLO]
[CODICE VERDE]
[CODICE BIANCO]
->Maria Neri

Test del costruttore di copia:
Numero pazienti: 3
[CODICE ROSSO]
[CODICE GIALLO]
[CODICE VERDE]
->Luigi Viola
[CODICE BIANCO]
->Maria Neri
->Carlo Bianchi

Test del distruttore:
(ps2 e' stato distrutto)
```

Note per la consegna:

Se nell'elaborato consegnato, compito.cpp **non compila**, l'elaborato è *insufficiente*.
Se nell'elaborato consegnato, compito.cpp **non collega** a main.cpp (prima parte), l'elaborato è *insufficiente*.
Se nell'elaborato consegnato, compito.cpp e main.cpp (prima parte) producono un programma che **non termina correttamente** (es., un errore di segmentazione) sul calcolatore su cui gira il portale di autocorrezione, l'elaborato è *insufficiente*.
Se nell'elaborato consegnato, il programma (prima parte) **non produce l'output atteso** ("scorretto"), l'elaborato è *insufficiente*.
Consiglio: per raggiungere la sufficienza, consigliamo di scrivere prima lo "scheletro" di tutte le funzioni chiamate dal main.cpp (prima parte), ovvero definirle con un corpo vuoto o contenente solo un'istruzione `return`, in modo da far compilare e collegare compito.cpp. Poi, consigliamo di programmare il corpo delle funzioni in modo da produrre l'output corretto. Successivamente, scrivere lo scheletro delle funzioni chiamate dal main.cpp (seconda parte), e programmarne il corpo.