Una `Mensa` rappresenta una mensa composta da un numero finito di sedie, che sono numerate a partire da 1. Ad ogni istante ciascuna sedia può essere libera oppure occupata da un pranzante, che è identificato attraverso una stringa di lunghezza qualsiasi.

Implementare le seguenti operazioni che possono essere effettuate su una `Mensa`:

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

- ✓ **`Mensa m(N);`**

Costruttore che inizializza una `Mensa` con `N` sedie. All'inizio tutte le sedie sono libere.

- ✓ **`cout << m;`**

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

```
sedia 1: Pranzante1
sedia 2: (non occupata)
sedia 3: (non occupata)
sedia 4: Pranzante2
```

L'output mostrato corrisponde a una mensa di 4 sedie, di cui la prima e la quarta occupata da pranzanti, rispettivamente "Pranzante1" e "Pranzante2". Le altre sedie sono libere. **Attenzione: deve essere presente un singolo spazio dopo i due punti e prima del nome del pranzante (o della stringa "(non occupata)")**.

- ✓ **`m.occupa(id);`**

Operazione che aggiunge il pranzante identificato dalla c-stringa `id` alla mensa `m`. Il nuovo pranzante viene accomodato nella prima sedia non occupata. Qualora la mensa sia piena o qualora quel pranzante sia già presente nella mensa, il metodo restituisce false, altrimenti true.

- ✓ **`m.libera(i);`**

Operazione che libera la sedia `i`-esima della mensa `m`. Qualora l'indice non sia valido, la mensa deve rimanere inalterata.

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

- ✓ **`~Mensa();`**

Qualora sia necessario, implementare il distruttore.

- ✓ **`Mensa m2 = m;`**

Costruttore di copia che costruisce `m2` utilizzando `m`.

- ✓ **`m.occupaGruppo(id, n);`**

Operazione che aggiunge un gruppo di `n` pranzanti identificato dalla c-stringa `id` ala mensa `m`. Nell'output e a tutti gli altri effetti, il gruppo di pranzanti equivale a `n` pranzanti singoli ma con sedie contigue e con lo stesso identificatore `id`. Il gruppo viene accomodato nella prima fila di `n` sedie contigue non occupate. Qualora non ci sia spazio nella mensa per `n` posti o che tali posti non siano contigui, o qualora l'identificatore del gruppo sia lo stesso di un pranzante già presente nella mensa, il metodo restituisce false, altrimenti true.

- ✓ **`m.ordina();`**

Operazione che sposta i pranzanti della mensa `m` tra le varie sedie in modo che compaiano in ordine alfabetico. Nel caso ci siano sedie vuote, queste verranno posizionate tutte nei primi indici della mensa, come se fossero prime nell'ordine alfabetico.

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

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
--- PRIMA PARTE ---
Test del costruttore e dell'operatore di uscita
sedia 1: (non occupata)
sedia 2: (non occupata)
sedia 3: (non occupata)
sedia 4: (non occupata)

Test della occupa
sedia 1: Pranzante1
sedia 2: (non occupata)
sedia 3: (non occupata)
sedia 4: (non occupata)

sedia 1: Pranzante1
sedia 2: Pranzante2
sedia 3: Pranzante3
sedia 4: Pranzante4

Test della libera (viene liberata la seconda sedia)
sedia 1: Pranzante1
sedia 2: (non occupata)
sedia 3: Pranzante3
sedia 4: Pranzante4

--- SECONDA PARTE ---
Test del costruttore di copia
sedia 1: Pranzante1
sedia 2: (non occupata)
sedia 3: Pranzante3
sedia 4: Pranzante4

Test dell'eventuale distruttore (m2 e' stata appena distrutta)
Test della ordina()
(prima dell'ordinamento)
sedia 1: Pranzante1
sedia 2: Pranzante0
sedia 3: Pranzante3
sedia 4: (non occupata)
(dopo l'ordinamento)
sedia 1: (non occupata)
sedia 2: Pranzante0
sedia 3: Pranzante1
sedia 4: Pranzante3
Test della occupaGruppo()
sedia 1: (non occupata)
sedia 2: Pranzante0
sedia 3: GruppoDiPranzo1
sedia 4: GruppoDiPranzo1
```

Note per la consegna:

Affinché l'elaborato venga considerato valido, il programma **deve** produrre almeno la prima parte dell'output atteso. In questo caso, i docenti procederanno alla valutazione dell'elaborato **solo se** lo studente avrà completato l'autocorrezione del proprio elaborato.
In **tutti** gli altri casi (per esempio, il programma non compila, non collega, non esegue o la prima parte dell'output non coincide con quella attesa), l'elaborato è considerato **insufficiente** e, pertanto, **non verrà corretto**.