Esame 20230120

Esercizio 3

(1) **Esercizio 3 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `collidiAsteroidi` che prende come parametri formali un array di interi `asteroidi`, la dimensione dell'array `numeroAsteroidi` e un numero passato per riferimento `numeroAsteroidiRimasti`. Ogni intero dell'array `asteroidi` rappresenta un asteroide: il valore assoluto dell'intero indica la dimensione dell'asteroide, mentre il segno indica la sua direzione (positivo indica destra, negativo indica sinistra). Ogni asteroide si muove alla stessa velocità.

Usando una pila come supporto, la funzione `collidiAsteroidi` deve ritornare un'array di interi allocato dinamicamente che contenga lo stato degli asteroidi dopo tutte le collisioni (`numeroAsteroidiRim` deve contenere il numero di elementi nell'array ritornato). Quando due asteroidi si scontrano, l'asteroide con la dimensione minore esplode. Se due asteroidi che si scontrano hanno la stessa dimensione, esplodono entrambi. Due asteroidi che si muovono nella stessa direzione non si scontreranno mai.
Questi sono quattro diversi esempi di esecuzione (notare come la stampa degli asteroidi rimasti—e quindi come l'array ritornato dalla funzione `collidiAsteroidi`—rispetta l'ordine degli asteroidi originali nell'array in input):

```
computer > ./a.out
array di asteroidi: {3, 6, -3}
asteroidi rimasti:  {3, 6}

computer > ./a.out
array di asteroidi: {7, -7}
asteroidi rimasti:  {}

computer > ./a.out
array di asteroidi: {-6, 10, 5, 8, -9}
asteroidi rimasti:  {-6, 10}

computer > ./a.out
array di asteroidi: {11, -5, -12, 4, -6}
asteroidi rimasti:  {-12, -6}
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `collidiAsteroidi` e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Scaricare anche i file `pila.cc` e `pila.h` i quali implementano le funzionalitá di una pila. Usare questi file nella risoluzione dell'esercizio;
- Ricordarsi di deallocare la memoria allocata dinamicamente e invocare la funzione `deinit()` della pila;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- È consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema;

- All'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`.

pila.cc
pila.h
esercizio3.cc

Information for graders:

(2) **Esercizio 3 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `collidiAsteroidi` che prende come parametri formali un array di interi `asteroidi`, la dimensione dell'array `numeroAsteroidi` e un numero passato per riferimento `numeroAsteroidiRimasti`. Ogni intero dell'array `asteroidi` rappresenta un asteroide: il valore assoluto dell'intero indica la dimensione dell'asteroide, mentre il segno indica la sua direzione (positivo indica sinistra, negativo indica destra). Ogni asteroide si muove alla stessa velocità.

Usando una pila come supporto, la funzione `collidiAsteroidi` deve ritornare un'array di interi allocato dinamicamente che contenga lo stato degli asteroidi dopo tutte le collisioni (`numeroAsteroidiRim` deve contenere il numero di elementi nell'array ritornato). Quando due asteroidi si scontrano, l'asteroide con la dimensione minore esplode. Se due asteroidi che si scontrano hanno la stessa dimensione, esplodono entrambi. Due asteroidi che si muovono nella stessa direzione non si scontreranno mai.
Questi sono quattro diversi esempi di esecuzione (notare come la stampa degli asteroidi rimasti—e quindi come l'array ritornato dalla funzione `collidiAsteroidi`—rispetta l'ordine degli asteroidi originali nell'array in input):

```
computer > ./a.out
array di asteroidi: {-3, -6, 3}
asteroidi rimasti:  {-3, -6}

computer > ./a.out
array di asteroidi: {-7, 7}
asteroidi rimasti:  {}

computer > ./a.out
array di asteroidi: {6, -10, -5, -8, 9}
asteroidi rimasti:  {6, -10}

computer > ./a.out
array di asteroidi: {-11, 5, 12, -4, 6}
asteroidi rimasti:  {12, 6}
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `collidiAsteroidi` e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Scaricare anche i file `pila.cc` e `pila.h` i quali implementano le funzionalitá di una pila. Usare questi file nella risoluzione dell'esercizio;
- Ricordarsi di deallocare la memoria allocata dinamicamente e invocare la funzione `deinit()` della pila;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- È consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema;
- All'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`.

pila.cc
pila.h
esercizio3.cc

Information for graders:

(3) **Esercizio 3 v3**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `collidiAsteroidi` che prende come parametri formali un array di interi `asteroidi`, la dimensione dell'array `numeroAsteroidi` e un numero passato per riferimento `numeroAsteroidiRimasti`. Ogni intero dell'array `asteroidi` rappresenta un asteroide: il valore assoluto dell'intero indica la dimensione dell'asteroide, mentre il segno indica la sua direzione (positivo indica destra, negativo indica sinistra). Ogni asteroide si muove alla stessa velocità.

Usando una pila come supporto, la funzione `collidiAsteroidi` deve ritornare un'array di interi allocato dinamicamente che contenga lo stato degli asteroidi dopo tutte le collisioni (`numeroAsteroidiRim` deve contenere il numero di elementi nell'array ritornato). Quando due asteroidi si scontrano, l'asteroide con la dimensione maggiore esplode. Se due asteroidi che si scontrano hanno la stessa dimensione, esplodono entrambi. Due asteroidi che si muovono nella stessa direzione non si scontreranno mai.
Questi sono quattro diversi esempi di esecuzione (notare come la stampa degli asteroidi rimasti—e quindi come l'array ritornato dalla funzione `collidiAsteroidi`—rispetta l'ordine degli asteroidi originali nell'array in input):

```
computer > ./a.out
array di asteroidi: {3, 6, -4}
asteroidi rimasti:  {3}

computer > ./a.out
array di asteroidi: {7, -7}
asteroidi rimasti:  {}

computer > ./a.out
array di asteroidi: {-4, 1, 5, 3, -2}
asteroidi rimasti:  {-4, 1}

computer > ./a.out
array di asteroidi: {2, -4, -1, 5, -3}
asteroidi rimasti:  {-1, -3}
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `collidiAsteroidi` e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Scaricare anche i file `pila.cc` e `pila.h` i quali implementano le funzionalitá di una pila. Usare questi file nella risoluzione dell'esercizio;
- Ricordarsi di deallocare la memoria allocata dinamicamente e invocare la funzione `deinit()` della pila;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- È consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema;
- All'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`.

pila.cc
pila.h
esercizio3.cc

Information for graders:

(4) **Esercizio 3 v4**

ESSAY | marked out of 10 | penalty 0 | File picker

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `collidiAsteroidi` che prende come parametri formali un array di interi `asteroidi`, la dimensione dell'array `numeroAsteroidi` e un numero passato per riferimento `numeroAsteroidiRimasti`. Ogni intero dell'array `asteroidi` rappresenta un asteroide: il valore assoluto dell'intero indica la dimensione dell'asteroide, mentre il segno indica la sua direzione (positivo indica sinistra, negativo indica destra). Ogni asteroide si muove alla stessa velocità.

Usando una pila come supporto, la funzione `collidiAsteroidi` deve ritornare un'array di interi allocato dinamicamente che contenga lo stato degli asteroidi dopo tutte le collisioni (`numeroAsteroidiRim` deve contenere il numero di elementi nell'array ritornato). Quando due asteroidi si scontrano, l'asteroide con la dimensione maggiore esplode. Se due asteroidi che si scontrano hanno la stessa dimensione, esplodono entrambi. Due asteroidi che si muovono nella stessa direzione non si scontreranno mai.
Questi sono quattro diversi esempi di esecuzione (notare come la stampa degli asteroidi rimasti—e quindi come l'array ritornato dalla funzione `collidiAsteroidi`—rispetta l'ordine degli asteroidi originali nell'array in input):

```
computer > ./a.out
array di asteroidi: {-3, -6, 4}
asteroidi rimasti:  {-3}

computer > ./a.out
array di asteroidi: {-7, 7}
asteroidi rimasti:  {}

computer > ./a.out
array di asteroidi: {4, -1, -5, -3, 2}
asteroidi rimasti:  {4, -1}

computer > ./a.out
array di asteroidi: {-2, 4, 1, -5, 3}
asteroidi rimasti:  {1, 3}
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `collidiAsteroidi` e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Scaricare anche i file `pila.cc` e `pila.h` i quali implementano le funzionalitá di una pila. Usare questi file nella risoluzione dell'esercizio;
- Ricordarsi di deallocare la memoria allocata dinamicamente e invocare la funzione `deinit()` della pila;
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione;
- È consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema;
- All'interno di questo programma non è ammesso l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`.

pila.cc
pila.h
esercizio3.cc

Information for graders:

Total of marks: 40