ASD a.a. 2023/24 - Esame del 9 gennaio 2025

Prima di cominciare lo svolgimento leggete attentamente tutto il testo.

Questa prova è organizzata in tre esercizi.
Vi forniamo un file zip che contiene per ogni esercizio: un file per completare la funzione da scrivere e un programma principale per lo svolgimento di test specifici per quella funzione. Ad esempio, per l'esercizio 1, saranno presenti un file es1.cpp e un file es1-test.o. Per compilare dovrete eseguire `g++ -std=c++11 -Wall es1.cpp es1-test.o -o es1-test`. E per eseguire il test, `./es1-test`. Dovete lavorare solo sui file indicati in ciascuno esercizio. Modificare gli altri file è sbagliato (ovviamente a meno di errata corrige indicata dai docenti).
In questi file dovete implementare le funzioni richieste, esattamente con la *segnatura* con cui sono indicate: nome, tipo restituito, tipo degli argomenti nell'ordine in cui sono dati. Non è consentito modificare queste informazioni. Potete invece fare quello che volete all'interno del corpo delle funzioni: in particolare, se contengono già una istruzione `return`, questa è stata inserita provvisoriamente per rendere compilabili i file ancora vuoti, e **dovrete modificarla in modo appropriato**.
Potete inoltre realizzare altre funzioni in tutti i casi in cui lo ritenete appropriato. Potete inserirvi tutti gli `#include` che vi servono oltre a quello relativo allo header con le funzioni da implementare. Attenzione però che **usare una funzione di libreria per evitare di scrivere del codice richiesto viene contato come errore** (esempio: se è richiesto di scrivere una funzione di ordinamento, usare la funzione `std::sort()` dal modulo di libreria standard `algorithm` è un errore).
Per ciascuno esercizio, vi diamo uno programma principale, che esegue i test. Controllate durante l'esecuzione del programma, quanti sono i test che devono essere superati e controllate l'esito (se non ci sono errori deve essere `SI` per tutti).

NB1: soluzioni particolarmente inefficienti potrebbero non ottenere la valutazione anche se forniscono i risultati attesi. Di contro ci riserviamo di premiare con un bonus soluzioni particolarmente ottimali.
NB2: superare positivamente tutti i test di una funzione non implica soluzione corretta e ottimale (e quindi valutazione massima).

1 Presentazione della struttura dati

L'obiettivo di questa prova di laboratorio è programmare tre funzioni per un tipo che memorizzi insiemi di coppie di numeri interi utilizzando liste doppiamente collegate. L'insieme vuoto sarà rappresentato da `nullptr`. Al contrario, un insieme non vuoto sarà rappresentato da una lista principale doppiamente collegata, in cui ogni nodo punta a una lista secondaria, anch'essa doppiamente collegata, che contiene coppie di numeri interi. Due coppie di interi appartengono alla stessa lista secondaria se la loro somma è uguale. Questa somma è memorizzata nel nodo della lista principale che punta alla lista secondaria corrispondente.
IMPORTANTE: Ogni nodo della lista principale punta necessariamente a una lista secondaria doppiamente collegata non vuota. Ogni coppia è presente una sola volta in una lista secondaria. Nella lista principale non possono esistere due nodi con lo stesso valore di somma. L'ordine degli elementi nelle liste, sia principali che secondarie, non è rilevante.
Alla fine di questo documento forniremo degli esempi di insieme. In questi esempi, se un puntatore punta a `nullptr`, esso non sarà rappresentato.

- La lista `li1` rappresenta l'insieme di tre coppie `{(2,1),(0,3),(3,0)}` che hanno tutti la stessa somma `3`.
- La lista `li2` rappresenta l'insieme di quattro coppie `{(1,2),(2,1),(0,3),(3,0)}`.
- La lista `li3` rappresenta l'insieme di cinque coppie `{(3,1),(1,2),(2,1),(0,3),(3,0)}`.
- La lista `li4` rappresenta l'insieme di sei coppie `{(-5,6),(3,1),(1,2),(2,1),(0,3),(3,0)}`.
- La lista `li5` rappresenta l'insieme di cinque coppie `{(-5,6),(1,2),(2,1),(0,3),(3,0)}`.

Nel file `set-dll.h` troverete la descrizione della struttura dati e i prototipi delle tre funzioni da implementare. **Non è consentito modificare questo file!**. Questo file è strutturato nel seguente modo:

```
#ifndef SET_DLL_H
#define SET_DLL_H

struct pair_node{
  int v1;
  int v2;
  pair_node* next_pair;
  pair_node* prev_pair;
};


struct set_node{
  int sum;
```

```
  pair_node* pairs;
  set_node* next;
  set_node* prev;
};

typedef set_node* set_list;

typedef pair_node* pair_list;

const set_list empty_set_list=nullptr;

/***********************************************/
/* Funzione da implementare                    */
/***********************************************/
//Es 1
//Ritorna il numero di coppie nell'insieme
unsigned int nbPairs(const set_list&);

//Es 2
//Aggiunge una coppia (v1,v2) all'insieme
//Se la coppia e' gia' presente, non fa nulla
void addPair(set_list&,int v1,int v2);

//Es 3
//Cancella dell'insieme la coppia (v1,v2)
//Se la coppia non e' presente non fa nulla
void deletePair(set_list&,int v1,int v2);
#endif
```

2 Esercizio 1

Nel file `es1.cpp`, dovete implementare la funzione `unsigned int nbPairs(const set_list& se)`. Questa funzione deve restituire il numero totale di coppie presenti nell'insieme `se`.
Esempi con gli insiemi dati alla fine di questo documento:

- `nbPairs(li1)` => 3
- `nbPairs(li2)` => 4
- `nbPairs(li3)` => 5
- `nbPairs(li4)` => 6
- `nbPairs(li5)` => 5

Per testare questa funzione, potete usare il file `es1-test.o` compilando con il comando:
`g++ -std=c++11 -Wall es1.cpp es1-test.o -o es1-test`.

3 Esercizio 2

Nel file `es2.cpp`, dovete implementare la funzione `void addPair(set_list& se, int v1, int v2);`. Questa funzione aggiunge la coppia `(v1, v2)` all'insieme se non è già presente. Se la coppia è già presente, la funzione non modifica l'insieme e non esegue alcuna azione.
Esempi con gli insiemi dati alla fine di questo documento:

- `addPair(li1,1,2)` cambia `li1` in `li2`
- `addPair(li2,3,1)` cambia `li2` in `li3`
- `addPair(li2,0,3)` non cambia `li2`

Per testare questa funzione, potete usare il file `es2-test.o` compilando con il comando:
`g++ -std=c++11 -Wall es2.cpp es2-test.o -o es2-test`.

4 Esercizio 3

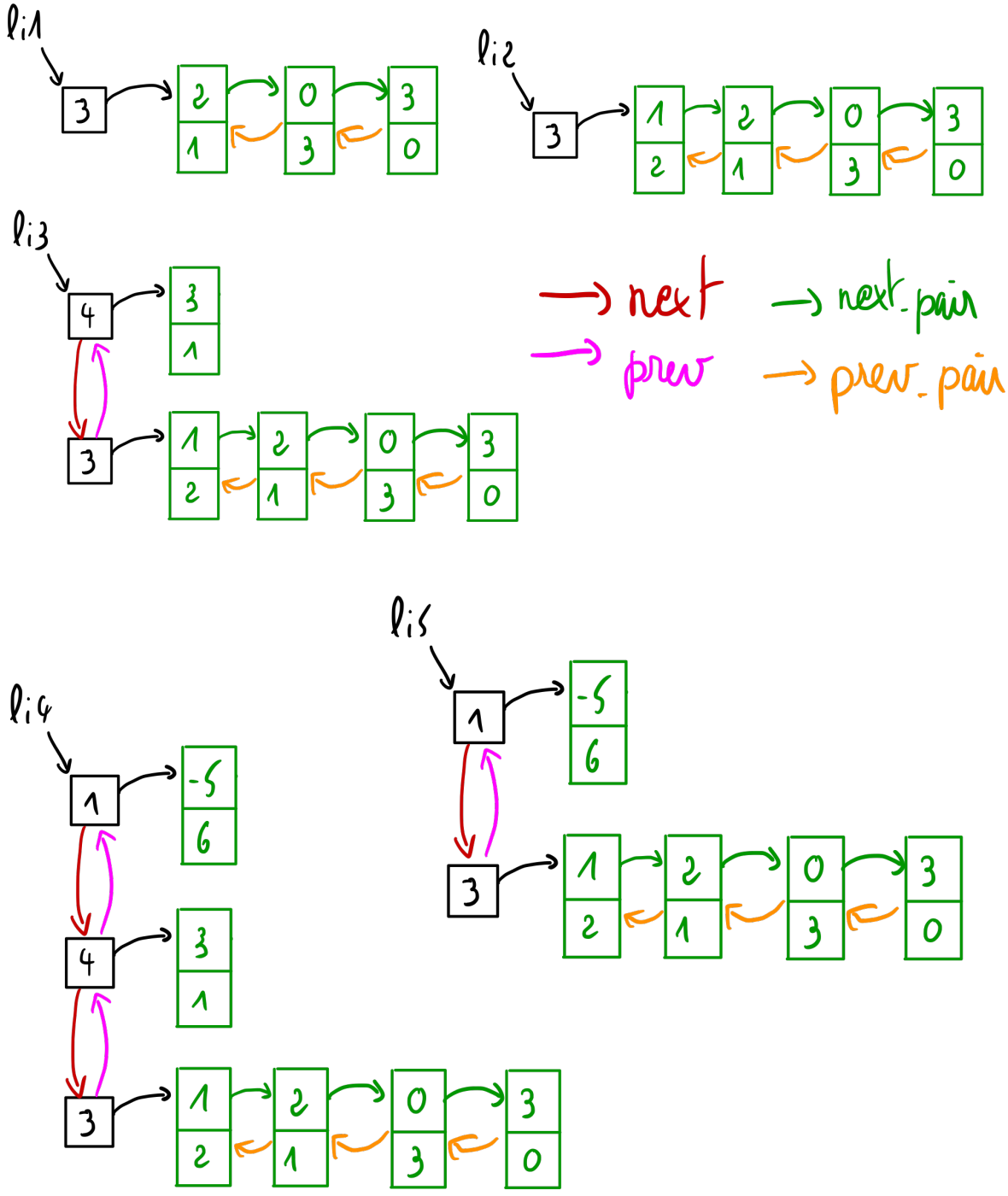
Nel file `es3.cpp`, dovete implementare la funzione `void deletePair(set_list& se, int v1, int v2);`. Questa funzione rimuove la coppia `(v1, v2)` dall'insieme se è presente. Se la coppia non è presente, la funzione non modifica l'insieme e non esegue alcuna azione. Attenzione a rispettare la forma della struttura, in particolare non c'è lista secondaria vuota.
Esempi con gli insiemi dati alla fine di questo documento:

- `deletePair(li2,1,2)` cambia `li2` in `li1`
- `deletePair(li3,3,1)` cambia `li3` in `li2`
- `deletePair(li3,6,1)` non cambia `li3`
- `deletePair(li4,3,1)` cambia `li4` in `li5`

Per testare questa funzione, potete usare il file `es3-test.o` compilando con il comando:
`g++ -std=c++11 -Wall es3.cpp es3-test.o -o es3-test`.

5 Consegna

Per la consegna, creare uno zip con tutti i file forniti.


li1
3
2
1
0
3
3
0
li2
3
1
2
2
1
0
3
3
0
li3
4
3
1
3
1
2
2
1
0
3
3
0
→ next
→ prev
→ next_pair
→ prev_pair
li4
1
-5
6
4
3
1
3
1
2
2
1
0
3
3
0
li5
1
-5
6
3
1
2
2
1
0
3
3
0