AND a.a. 2024/25 - Esame del 15 Luglio 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).

NB [PATTO]: gli studenti che hanno sottoscritto il patto devono svolgere solo l'esercizio 1 e 2.
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

Lo scopo di questo laboratorio è di implementare delle funzioni per una struttura dati denominata coda con priorità. La coda con priorità è rappresentata da una lista semplice dove ogni nodo contiene un intero rappresentante una priorità e un puntatore verso una lista circolare doppiamente collegata **con sentinella** che contiene delle string. La coda con priorità vuota è rappresentata da `nullptr`. Intuitivamente, quando una nuova string `st` con priorità `p` deve essere inserita nella coda allora `st` è inserita in coda alla lista circolare puntata dal nodo della lista semplice contenente il valore `p` (se non esiste, questo nodo e la lista circolare associata vengono creati). Per una lista circolare con sentinella, chiamiamo coda il nodo indicato dalla sentinella con il puntatore `prev2`. Per estrarre un elemento dalla coda, prendiamo il più vecchio elemento con la priorità più piccola, detto in altro modo, se `pmin` è la priorità più piccola, si estrae la string in testa dalla lista circolare puntata dal nodo della lista semplice con valore `pmin`. Per una lista circolare con sentinella, chiamiamo testa il nodo indicato dalla sentinella con il puntatore `next2`.
In più, la coda con priorità deve rispettare le seguente regole:

- nella lista semplice, i nodi sono **ordinati con priorità crescente** e non ci sono due nodi con la stessa priorità;
- ogni nodo della lista semplice, ha un puntatore verso una lista circolare doppiamente collegata con sentinella **non vuota**;
- una string può essere presente più volte nella coda con priorità e anche più volte nella stessa lista circolare;
- la struttura dati non contiene string vuote (uguale a `""`).

Nel file `priority-circ.h` troverete la descrizione della struttura dati e i prototipi delle tre funzioni da implementare. **Non dovete modificare questo file!**. Questo file contiene:

```
struct dllCell {
  String val;
  dllCell *next2;
  dllCell *prev2;
};

struct cell{
  int priority;
  dllCell* queue;
  cell *next;
};

typedef dllCell *dll;
typedef cell *list;
```

```
const list emptyList=nullptr;

/*********************************************/
/* Funzione da implementare                  */
/*********************************************/

//Es 1
//Ritorna il numero di elementi (string) nella coda
unsigned int nbElem(const list&);


//Es 2
//Inserisce una string con priorita' nella coda
//Se la string da inserire e' uguale a "" non fa nulla
void insertElem(list&,std::string,int);

//Es 3
//Ritorna l'elemento piu' vecchio con la priorita' piu' bassa della coda
//Se la coda e' vuota, ritorna la string vuota ""
std::string removeFirst(list&);
```

Alla fine di questo documento trovate degli esempi di coda con priorità.

2 Esercizio 1

Nel file `es1.cpp`, dovete implementare la funzione `unsigned int nbElem(const list& li)`. Questa funzione ritorna il numero di elementi (string) nella coda con priorità `li`.
Esempi di esecuzione della funzione sulle code date alla fine di questo documento:

- `nbElem(li1)` => 1
- `nbElem(li2)` => 2
- `nbElem(li3)` => 3
- `nbElem(li4)` => 4
- `nbElem(li5)` => 3
- `nbElem(li6)` => 3
- `nbElem(li7)` => 4

Per testare questa funzione, potete usare il file `es1-test.o` compilando con l'istruzione
`g++ -std=c++11 -Wall es1.cpp es1-test.o -o es1-test`.
In questi test, quando stampiamo una coda, stampiamo le priorità come sono ordinate nella lista semplice e per ogni priorità stampiamo il contenuto dalla lista circolare dalla testa alla coda. Ad esempio, la stampa dell'albero `li7` dato alla fine di questo documento è la seguente:

```
3: A <-> C <-> C
7: A
nullptr
```

3 Esercizio 2

Nel file `es2.cpp`, dovete implementare la funzione `void insertElem(list& li,std::string st,int p)`. Questa funzione inserisce in coda alla lista circolare con priorità `p` la string `st` (la coda della lista circolare essendo il nodo indicata dal campo `prev2` dalla sentinella). Se non esiste nella coda una lista circolare con priorità `p`, la funzione la crea (e la mette alla posizione giusta sapendo che le priorità sono ordinate in ordine crescente nella lista semplice). Se la string `st` è uguale a `""`, la funzione non fa nulla.
Esempi di esecuzione della funzione con le code date alla fine di questo documento:

- Se `li0=nullptr`, allora `insertElem(li0,"A",3)` trasforma `li0` in `li1`
- `insertElem(li1,"E",1)` trasforma `li1` in `li2`

- `insertElem(li2,"C",3)` trasforma `li2` in `li3`
- `insertElem(li3,"D",1)` trasforma `li3` in `li4`
- `insertElem(li6,"A",7)` trasforma `li6` in `li7`
- `insertElem(li7,"Z",5)` trasforma `li7` in `li8`

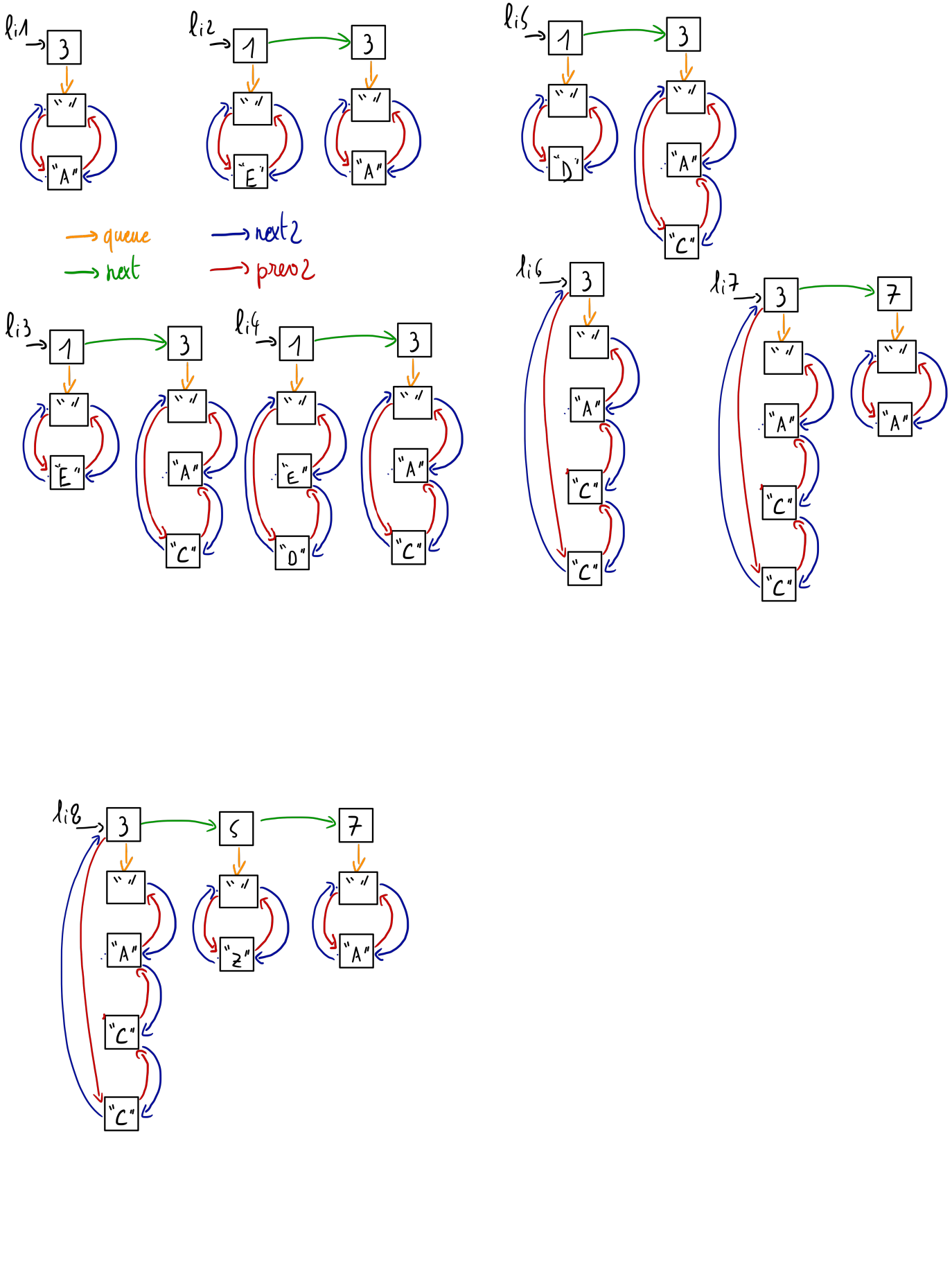
Per testare questa funzione, potete usare il file `es2-test.o` compilando con l'istruzione:
`g++ -std=c++11 -Wall es2.cpp es2-test.o -o es2-test`.

4 Esercizio 3

Nel file `es3.cpp`, dovete implementare la funzione `std::string removeFirst(list& li)`. Questa funzione rimuove dalla coda l'elemento più vecchio con la priorità più piccola e lo ritorna. In pratica, rimuove l'elemento in testa della lista circolare con la priorità più piccola (ricordiamo che la testa è il nodo indicato con il campo `next2` dalla sentinella). Se la lista circolare modificata diventa vuota, si deve anche togliere il nodo con la priorità corrispondente dalla lista semplice. Se la coda è vuota, la funziona ritorna la string vuota `""`.
Esempi di esecuzione della funzione con le code date alla fine di questo documento:

- `removeFist(li0)` con `li0` uguale a `nullptr` non cambia `li0` e ritorna `""`
- `removeFist(li1)` cambia `li1` in `nullptr` in e ritorna `"A"`
- `removeFist(li2)` cambia `li2` in `li1` in e ritorna `"E"`
- `removeFist(li4)` cambia `li4` in `li5` in e ritorna `"E"`

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


li1
3
""
"A"
li2
1
3
""
"E"
""
"A"
queue
next
next2
prev2
li3
1
3
""
"E"
""
"A"
"C"
li4
1
3
""
"E"
"D"
""
"A"
"C"
li5
1
3
""
"D"
""
"A"
"C"
li6
3
""
"A"
"C"
"C"
li7
3
7
""
"A"
"C"
"C"
""
"A"
li8
3
5
7
""
"A"
"C"
"C"
""
"Z"
""
"A"