#### Esame 20220111

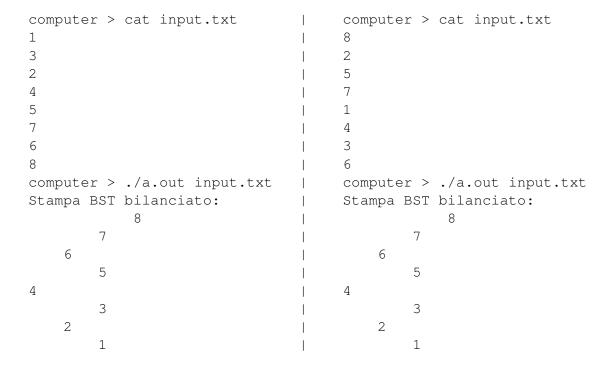
# Esercizio 3

#### (1) Esercizio 3 v1



Scrivere nel file esercizio3.cc un programma che riceve da linea di comando (quindi usando argc e argv) il percorso di un file contenente un numero intero per riga. Il file contiene almeno 1 numero intero e al massimo 100 interi, e tutti gli interi sono diversi fra loro. Il programma deve creare un albero di ricerca binaria **bilanciato** con i numeri contenuti nel file.

Questi sono due esempi di esecuzione:



# Note:

- Scaricare il file esercizio3.cc, modificarlo per implementare le funzionalità richieste e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Implementare controlli sulla buona riuscita delle operazioni (e.g., l'apertura del file di input) e stampare a video messaggi d'errore appropriati;
- E' consentito definire e 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 e fstream.
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione.

# Suggerimenti:

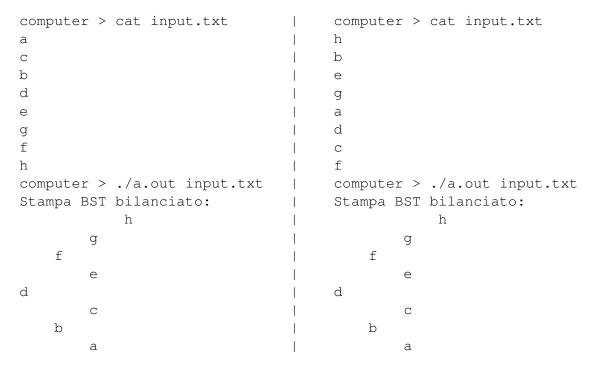
- Ricordarsi che l'albero di ricerca binaria dev'essere **bilanciato**. Quindi, il primo elemento contenuto nel file di input non dev'essere necessariamente la radice dell'albero. Gli elementi contenuti nel file di input non sono necessariamente ordinati.
- Il file esercizio3.cc contiene già una funzione stampaAlbero che stampa a terminale l'albero evidenziando le relazioni fra i nodi. Potete utilizzare la funzione stampaAlbero per controllare se l'albero è bilanciato.

### (2) Esercizio 3 v2



Scrivere nel file esercizio3.cc un programma che riceve da linea di comando (quindi usando argc e argv) il percorso di un file contenente una lettera dell'alfabeto per riga (26 lettere). Il file contiene almeno 1 lettera e al massimo 26 lettere, e tutte le lettere sono diverse fra loro. Il programma deve creare un albero di ricerca binaria **bilanciato** con le lettere contenute nel file, ordinate alfabeticamente.

Questi sono due esempi di esecuzione:



#### Note:

- Scaricare il file esercizio3.cc, modificarlo per implementare le funzionalità richieste e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Implementare controlli sulla buona riuscita delle operazioni (e.g., l'apertura del file di input) e stampare a video messaggi d'errore appropriati;
- E' consentito definire e 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 e fstream.
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione.

#### Suggerimenti:

• Ricordarsi che l'albero di ricerca binaria dev'essere **bilanciato**. Quindi, il primo elemento contenuto nel file di input non dev'essere necessariamente la radice dell'albero. Gli elementi contenuti nel file di input non sono necessariamente ordinati.

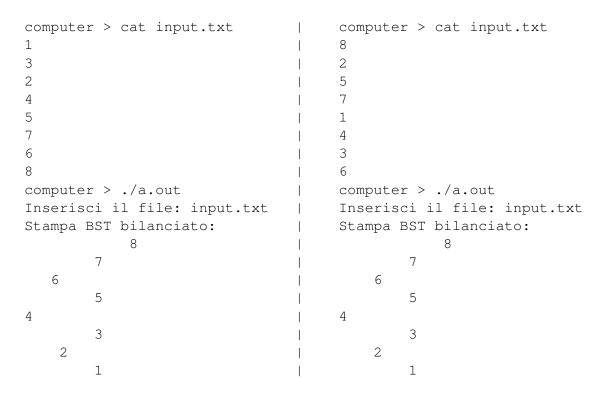
• Il file esercizio3.cc contiene già una funzione stampaAlbero che stampa a terminale l'albero evidenziando le relazioni fra i nodi. Potete utilizzare la funzione stampaAlbero per controllare se l'albero è bilanciato.

# (3) Esercizio 3 v3



Scrivere nel file esercizio3.cc un programma che riceve da console input (quindi da cin) il percorso di un file (il percorso è lungo al massimo 30 caratteri). Il file contiene un numero intero per riga. Il file contiene almeno 1 numero intero e al massimo 100 interi, e tutti gli interi sono diversi fra loro. Il programma deve creare un albero di ricerca binaria **bilanciato** con i numeri contenuti nel file.

Questi sono due esempi di esecuzione:



#### Note:

- Scaricare il file esercizio3.cc, modificarlo per implementare le funzionalità richieste e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Implementare controlli sulla buona riuscita delle operazioni (e.g., l'apertura del file di input) e stampare a video messaggi d'errore appropriati;
- E' consentito definire e 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.
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione.

#### Suggerimenti:

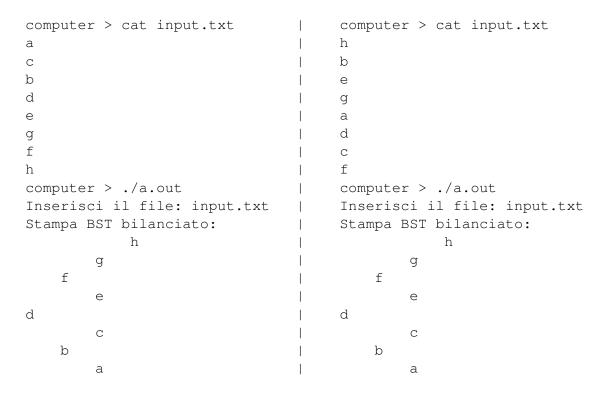
- Ricordarsi che l'albero di ricerca binaria dev'essere **bilanciato**. Quindi, il primo elemento contenuto nel file di input non dev'essere necessariamente la radice dell'albero. Gli elementi contenuti nel file di input non sono necessariamente ordinati.
- Il file esercizio3.cc contiene già una funzione stampaAlbero che stampa a terminale l'albero evidenziando le relazioni fra i nodi. Potete utilizzare la funzione stampaAlbero per controllare se l'albero è bilanciato.

#### (4) Esercizio 3 v4



Scrivere nel file esercizio3.cc un programma che riceve da console input (quindi da cin) il percorso di un file (il percorso è lungo al massimo 30 caratteri). Il file contiene una lettera dell'alfabeto per riga (26 lettere). Il file contiene almeno 1 lettera e al massimo 26 lettere, e tutte le lettere sono diverse fra loro. Il programma deve creare un albero di ricerca binaria **bilanciato** con le lettere contenute nel file, ordinate alfabeticamente.

Questi sono due esempi di esecuzione:



#### Note:

- Scaricare il file esercizio3.cc, modificarlo per implementare le funzionalità richieste e infine caricare il file risultato delle vostre modifiche a soluzione di questo esercizio nello spazio apposito;
- Ricordarsi di deallocare la memoria allocata dinamicamente;
- Implementare controlli sulla buona riuscita delle operazioni (e.g., l'apertura del file di input) e stampare a video messaggi d'errore appropriati;
- E' consentito definire e 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.
- Ricordarsi di distinguere gli esempi nella descrizione dell'esercizio (che servono solo ad aiutare a comprendere il problema) dalle istruzioni di implementazione.

#### Suggerimenti:

- Ricordarsi che l'albero di ricerca binaria dev'essere **bilanciato**. Quindi, il primo elemento contenuto nel file di input non dev'essere necessariamente la radice dell'albero. Gli elementi contenuti nel file di input non sono necessariamente ordinati.
- Il file esercizio3.cc contiene già una funzione stampaAlbero che stampa a terminale l'albero evidenziando le relazioni fra i nodi. Potete utilizzare la funzione stampaAlbero per controllare se l'albero è bilanciato.

Total of marks: 40