1 Esercizio 3_1

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` che prende come parametri formali un array di caratteri `espressione` e, tramite riferimento, una variabile booleana `corretta`. L'array `espressione` contiene un'espressione booleana in notazione prefissa, cioè dove gli operatori si trovano a sinistra degli argomenti. Per esempio, al posto di scrivere 'T&F', si scrive '&TF'. Inoltre, l'espressione si legge da destra a sinistra. Per esempio, l'espressione '&F&TT' è equivalente a '&F(&TT). L'espressione booleana è composta da una combinazione dei soli seguenti quattro caratteri:

- `T` che rappresenta il valore "True";
- `F` che rappresenta il valore "False";
- `&` che rappresenta l'operatore `AND` logico;
- `!` che rappresenta l'operatore `NOT` logico.

La funzione `parseEspressioneBooleanaPrefissa` deve ritornare un carattere (`T` oppure `F`) corrispondente alla valutazione dell'espressione. Se l'espressione non è ben formata, indicarlo attraverso la variabile booleana `corretta`. Ricordare che l'operatore `!` ha la precedenza sull'operatore `&`.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out
Inserisci l'espressione: &T!T
Risultato: F
```

```
computer > ./a.out
Inserisci l'espressione: &F&TT
Risultato: F
```

```
computer > ./a.out
Inserisci l'espressione: &!F!F
Risultato: T
```

```
computer > ./a.out
Inserisci l'espressione: F&T
Espressione malformata
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` 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 funzionalità di una pila. Usare questi file nella risoluzione dell'esercizio. Ricordarsi di deallocare la memoria;
- 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`.

2 Esercizio 3_2

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` che prende come parametri formali un array di caratteri `espressione` e, tramite riferimento, una variabile booleana `corretta`. L'array `espressione` contiene un'espressione booleana in notazione prefissa, cioè dove gli operatori si trovano a sinistra degli argomenti. Per esempio, al posto di scrivere 'T|F', si scrive '|TF'. Inoltre, l'espressione si legge da destra a sinistra. Per esempio, l'espressione '|F|TT' è equivalente a '|F(|TT). L'espressione booleana è composta da una combinazione dei soli seguenti quattro caratteri:

- `T` che rappresenta il valore "True";
- `F` che rappresenta il valore "False";
- | che rappresenta l'operatore `OR` logico;
- `!` che rappresenta l'operatore `NOT` logico.

La funzione `parseEspressioneBooleanaPrefissa` deve ritornare un carattere (`T` oppure `F`) corrispondente alla valutazione dell'espressione. Se l'espressione non è ben formata, indicarlo attraverso la variabile booleana `corretta`. Ricordare che l'operatore `!` ha la precedenza sull'operatore |.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out
Inserisci l'espressione: |T!T
Risultato: T

computer > ./a.out
Inserisci l'espressione: |!T!T
Risultato: F
```

```
computer > ./a.out
Inserisci l'espressione: |F|F!F
Risultato: T

computer > ./a.out
Inserisci l'espressione: F|T
Espressione malformata
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` 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 funzionalità di una pila. Usare questi file nella risoluzione dell'esercizio. Ricordarsi di deallocare la memoria;
- 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`.

3 Esercizio 3_3

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` che prende come parametri formali un array di caratteri `espressione` e, tramite riferimento, una variabile booleana `corretta`. L'array `espressione` contiene un'espressione booleana in notazione prefissa, cioè dove gli operatori si trovano a sinistra degli argomenti. Per esempio, al posto di scrivere '1&0', si scrive '&10'. Inoltre, l'espressione si legge da destra a sinistra. Per esempio, l'espressione '&0&11' è equivalente a '&0(&11). L'espressione booleana è composta da una combinazione dei soli seguenti quattro caratteri:

- `1` che rappresenta il valore "True";
- `0` che rappresenta il valore "False";
- `&` che rappresenta l'operatore `AND` logico;
- `!` che rappresenta l'operatore `NOT` logico.

La funzione `parseEspressioneBooleanaPrefissa` deve ritornare un carattere (`1` oppure `0`) corrispondente alla valutazione dell'espressione. Se l'espressione non è ben formata, indicarlo attraverso la variabile booleana `corretta`. Ricordare che l'operatore `!` ha la precedenza sull'operatore `&`.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out
Inserisci l'espressione: &1!1
Risultato: 0

computer > ./a.out
Inserisci l'espressione: &0&11
Risultato: 0

computer > ./a.out
Inserisci l'espressione: &!0!0
Risultato: 1

computer > ./a.out
Inserisci l'espressione: 0&1
Espressione malformata
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` 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 funzionalità di una pila. Usare questi file nella risoluzione dell'esercizio. Ricordarsi di deallocare la memoria;
- 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`.

4 Esercizio 3_4

Scrivere nel file `esercizio3.cc` la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` che prende come parametri formali un array di caratteri `espressione` e, tramite riferimento, una variabile booleana `corretta`. L'array `espressione` contiene un'espressione booleana in notazione prefissa, cioè dove gli operatori si trovano a sinistra degli argomenti. Per esempio, al posto di scrivere '1|0', si scrive '|10'. Inoltre, l'espressione si legge da destra a sinistra. Per esempio, l'espressione '|0|11' è equivalente a '|0(|11). L'espressione booleana è composta da una combinazione dei soli seguenti quattro caratteri:

- `1` che rappresenta il valore "True";
- `0` che rappresenta il valore "False";
- | che rappresenta l'operatore `OR` logico;
- `!` che rappresenta l'operatore `NOT` logico.

La funzione `parseEspressioneBooleanaPrefissa` deve ritornare un carattere (`1` oppure `0`) corrispondente alla valutazione dell'espressione. Se l'espressione non è ben formata, indicarlo attraverso la variabile booleana `corretta`. Ricordare che l'operatore `!` ha la precedenza sull'operatore |.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out
Inserisci l'espressione: |1!1
Risultato: 1
```

```
computer > ./a.out
Inserisci l'espressione: |0|0!0
Risultato: 1
```

```
computer > ./a.out
Inserisci l'espressione: |!1!1
Risultato: 0
```

```
computer > ./a.out
Inserisci l'espressione: 0|1
Espressione malformata
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `parseEspressioneBooleanaPrefissa` 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 funzionalità di una pila. Usare questi file nella risoluzione dell'esercizio. Ricordarsi di deallocare la memoria;
- 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`.