Esame 20220901

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 `decodificaNumero` che prende come parametro formale un array di caratteri `numeroCodificato`. L'array `numeroCodificato` contiene fino ad un massimo di 8 caratteri, scelti esclusivamente fra "I" e "D", e codifica un numero intero seguendo le seguenti regole:

- il carattere "I" indica che la cifra successiva deve essere maggiore di quella precedente;
- il carattere "D" indica che la cifra successiva deve essere minore di quella precedente;
- il numero formato dalla concatenazione delle cifre è un numero intero `int` composto solo da cifre fra 1 e 9 (escluso quindi lo 0) tutte diverse fra loro (le cifre non si possono ripetere);
- il numero formato dalla concatenazione delle cifre è il minore possibile. Per esempio, la sequenza "IIII" codifica il numero 12345, e non il numero 23456;

La funzione `decodificaNumero` deve ritornare il numero (come intero) codificato dall'array di caratteri `numeroCodificato`, e **deve utilizzare una pila** per costruire la decodifica.

Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out              computer > ./a.out
Inserisci l'array: IIDDIDID     Inserisci l'array: IDIDII
Decodifica:        125437698    Decodifica:        1325467
computer > ./a.out              computer > ./a.out
Inserisci l'array: DDDD         Inserisci l'array: IIII
Decodifica:        54321        Decodifica:        12345
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `decodificaNumero` 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 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`;

- All'interno di questo programma è ammesso l'uso delle funzioni della libreria `iostream` e delle funzioni `pow(base, esp)`—per il calcolo delle potenze—e `strlen(array)`—per il calcolo della lunghezza di un array di caratteri—definite nelle librerie `string.h` e `math.h`, rispettivamente.

Suggerimenti:

- Notare come una sequenza composta da `x` caratteri codifica un numero di `x+1` cifre. Si puo' quindi intuire che un carattere non corrisponde ad una cifra, ma mette in relazione la cifra precedente con quella successiva;
- La funzione `decodificaNumero` puo' essere implementata iterando sull'array `numeroCodificato` (da 0 alla lunghezza di `numeroCodificato`, estremi inclusi) e manipolando gli indici (i.e., le posizioni dei caratteri nell'array `numeroCodificato`) con la pila, seguendo il seguente pseudocodice (implementazioni diverse sono ovviamente possibili):

```
int numero = 0;
int lenCodifica = strlen(input);
int esponente = lenCodifica;
for (int i = 0; i <= lenCodifica; i++) {
    // inserisco i+1 nella pila
    // se il carattere corrente rappresenta un maggiore
    // oppure ho raggiunto l'ultimo carattere
    //     calcolo la cifra corrente svuotando lo stack,
    //     aggiornando le variabili esponente e numero
    ...
}
return numero;
```

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 `decodificaNumero` che prende come parametro formale un array di caratteri `numeroCodificato`. L'array `numeroCodificato` contiene fino ad un massimo di 8 caratteri, scelti esclusivamente fra "M" e "m", e codifica un numero intero seguendo le seguenti regole:

- il carattere "M" indica che la cifra successiva deve essere maggiore di quella precedente;
- il carattere "m" indica che la cifra successiva deve essere minore di quella precedente;
- il numero formato dalla concatenazione delle cifre è un numero intero `int` composto solo da cifre fra 1 e 9 (escluso quindi lo 0) tutte diverse fra loro (le cifre non si possono ripetere);
- il numero formato dalla concatenazione delle cifre è il minore possibile. Per esempio, la sequenza "MMMM" codifica il numero 12345, e non il numero 23456;

La funzione `decodificaNumero` deve ritornare il numero (come intero) codificato dall'array di caratteri `numeroCodificato`, e **deve utilizzare una pila** per costruire la decodifica.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out               computer > ./a.out
Inserisci l'array: MMmmMmMm      Inserisci l'array: MmMmMM
Decodifica:        125437698     Decodifica:        1325467
computer > ./a.out               computer > ./a.out
Inserisci l'array: mmmm          Inserisci l'array: MMMM
Decodifica:        54321         Decodifica:        12345
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `decodificaNumero` 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 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`;
- All'interno di questo programma è ammesso l'uso delle funzioni della libreria `iostream` e delle funzioni `pow(base, esp)`—per il calcolo delle potenze—e `strlen(array)`—per il calcolo della lunghezza di un array di caratteri—definite nelle librerie `string.h` e `math.h`, rispettivamente.

Suggerimenti:

- Notare come una sequenza composta da `x` caratteri codifica un numero di `x+1` cifre. Si puo' quindi intuire che un carattere non corrisponde ad una cifra, ma mette in relazione la cifra precedente con quella successiva;
- La funzione `decodificaNumero` puo' essere implementata iterando sull'array `numeroCodificato` (da 0 alla lunghezza di `numeroCodificato`, estremi inclusi) e manipolando gli indici (i.e., le posizioni dei caratteri nell'array `numeroCodificato`) con la pila, seguendo il seguente pseudocodice (implementazioni diverse sono ovviamente possibili):

```
int numero = 0;
int lenCodifica = strlen(input);
int esponente = lenCodifica;
for (int i = 0; i <= lenCodifica; i++) {
    // inserisco i+1 nella pila
    // se il carattere corrente rappresenta un maggiore
    // oppure ho raggiunto l'ultimo carattere
    //     calcolo la cifra corrente svuotando lo stack,
    //     aggiornando le variabili esponente e numero
    ...
}
return numero;
```

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 `decodificaNumero` che prende come parametro formale un array di caratteri `numeroCodificato`. L'array `numeroCodificato` contiene fino ad un massimo di 8 caratteri, scelti esclusivamente fra "+" e "-", e codifica un numero intero seguendo le seguenti regole:

- il carattere "+" indica che la cifra successiva deve essere maggiore di quella precedente;
- il carattere "-" indica che la cifra successiva deve essere minore di quella precedente;
- il numero formato dalla concatenazione delle cifre è un numero intero `int` composto solo da cifre fra 1 e 9 (escluso quindi lo 0) tutte diverse fra loro (le cifre non si possono ripetere);
- il numero formato dalla concatenazione delle cifre è il minore possibile. Per esempio, la sequenza "++++" codifica il numero 12345, e non il numero 23456;

La funzione `decodificaNumero` deve ritornare il numero (come intero) codificato dall'array di caratteri `numeroCodificato`, e **deve utilizzare una pila** per costruire la decodifica.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out              computer > ./a.out
Inserisci l'array: ++--+-+-     Inserisci l'array: +-+-++
Decodifica:        125437698    Decodifica:        1325467
computer > ./a.out              computer > ./a.out
Inserisci l'array: ----         Inserisci l'array: ++++
Decodifica:        54321        Decodifica:        12345
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `decodificaNumero` 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 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`;
- All'interno di questo programma è ammesso l'uso delle funzioni della libreria `iostream` e delle funzioni `pow(base, esp)`—per il calcolo delle potenze—e `strlen(array)`—per il calcolo della lunghezza di un array di caratteri—definite nelle librerie `string.h` e `math.h`, rispettivamente.

Suggerimenti:

- Notare come una sequenza composta da `x` caratteri codifica un numero di `x+1` cifre. Si puo' quindi intuire che un carattere non corrisponde ad una cifra, ma mette in relazione la cifra precedente con quella successiva;
- La funzione `decodificaNumero` puo' essere implementata iterando sull'array `numeroCodificato` (da 0 alla lunghezza di `numeroCodificato`, estremi inclusi) e manipolando gli indici (i.e., le posizioni dei caratteri nell'array `numeroCodificato`) con la pila, seguendo il seguente pseudocodice (implementazioni diverse sono ovviamente possibili):

```
int numero = 0;
int lenCodifica = strlen(input);
int esponente = lenCodifica;
for (int i = 0; i <= lenCodifica; i++) {
    // inserisco i+1 nella pila
    // se il carattere corrente rappresenta un maggiore
    // oppure ho raggiunto l'ultimo carattere
    //     calcolo la cifra corrente svuotando lo stack,
    //     aggiornando le variabili esponente e numero
    ...
}
return numero;
```

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 `decodificaNumero` che prende come parametro formale un array di caratteri `numeroCodificato`. L'array `numeroCodificato` contiene fino ad un massimo di 8 caratteri, scelti esclusivamente fra "A" e "B", e codifica un numero intero seguendo le seguenti regole:

- il carattere "A" indica che la cifra successiva deve essere maggiore di quella precedente;
- il carattere "B" indica che la cifra successiva deve essere minore di quella precedente;
- il numero formato dalla concatenazione delle cifre è un numero intero `int` composto solo da cifre fra 1 e 9 (escluso quindi lo 0) tutte diverse fra loro (le cifre non si possono ripetere);
- il numero formato dalla concatenazione delle cifre è il minore possibile. Per esempio, la sequenza "AAAA" codifica il numero 12345, e non il numero 23456;

La funzione `decodificaNumero` deve ritornare il numero (come intero) codificato dall'array di caratteri `numeroCodificato`, e **deve utilizzare una pila** per costruire la decodifica.
Questi sono quattro diversi esempi di esecuzione:

```
computer > ./a.out              computer > ./a.out
Inserisci l'array: AABBABAB     Inserisci l'array: ABABAA
Decodifica:        125437698    Decodifica:        1325467
computer > ./a.out              computer > ./a.out
Inserisci l'array: BBBB         Inserisci l'array: AAAA
Decodifica:        54321        Decodifica:        12345
```

Note:

- Scaricare il file `esercizio3.cc`, modificarlo per inserire la corretta implementazione della funzione `decodificaNumero` 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 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`;
- All'interno di questo programma è ammesso l'uso delle funzioni della libreria `iostream` e delle funzioni `pow(base, esp)`—per il calcolo delle potenze—e `strlen(array)`—per il calcolo della lunghezza di un array di caratteri—definite nelle librerie `string.h` e `math.h`, rispettivamente.

Suggerimenti:

- Notare come una sequenza composta da `x` caratteri codifica un numero di `x+1` cifre. Si puo' quindi intuire che un carattere non corrisponde ad una cifra, ma mette in relazione la cifra precedente con quella successiva;
- La funzione `decodificaNumero` puo' essere implementata iterando sull'array `numeroCodificato` (da 0 alla lunghezza di `numeroCodificato`, estremi inclusi) e manipolando gli indici (i.e., le posizioni dei caratteri nell'array `numeroCodificato`) con la pila, seguendo il seguente pseudocodice (implementazioni diverse sono ovviamente possibili):

```
int numero = 0;
int lenCodifica = strlen(input);
int esponente = lenCodifica;
for (int i = 0; i <= lenCodifica; i++) {
    // inserisco i+1 nella pila
    // se il carattere corrente rappresenta un maggiore
    // oppure ho raggiunto l'ultimo carattere
    //     calcolo la cifra corrente svuotando lo stack,
    //     aggiornando le variabili esponente e numero
    ...
}
return numero;
```

Information for graders:

Total of marks: 40