Algoritmi in C# per il controllo d'errore

Parità

1. [**FunzioniBit**] – Scrivere le funzioni **BitParitaPari** e **BitParitaDispari** che ricevono in ingresso un numero intero senza segno su 8 bit e restituiscono il valore del bit di parità (rispettivamente "pari" e "dispari"). Scrivere quindi un programma che utilizzi le funzioni per riportare entrambi i bit di parità dei numeri 65, 174 e del carattere 'G' (codifica ASCII).

2. [**Vcr1**] – Scrivere una funzione **Vcr** che riceva un intero N senza segno codificato su 8 bit e minore di 128 (il suo valore è quindi espremibile al massimo con 7 bit), calcoli il **bit di parità pari**, aggiungendolo al bit msb di N e restituendo il risultato. Per esempio, il numero 0110010 ha bit di parità "pari" uguale a **1:** il risultato finale è **1**0110010.
   Scrivere inoltre un programma che: richieda l'inserimento di una stringa di caratteri da tastiera; generi e stampi un vettore **v1** contenente i codici ASCII dei singoli caratteri; memorizzi successivamente la codifica VCR dei singoli caratteri in un vettore **v2**; stampi infine il contenuto di **v2**.
   Esempio: dalla stringa "NODI" si ricavano i seguenti ouput:

```
Codifica iniziale della stringa 'NODI':
   78       79       68       73
01001110 01001111 01000100 01001001

Codifica finale (algoritmo VCR)
   78      207       68      201
01001110 11001111 01000100 11001001
```

3. [**Vcr2**] – Scrivere un programma che simuli la ricezione di una sequenza binaria (rappresentata da un vettore di numeri) codificata mediante VCR (vedi programma **Vcr1**) e verifichi l'integrità del messaggio contenuto. Il programma riutilizza le funzioni definite nei problemi precedenti per analizzare un vettore di numeri (costanti), riportando la frase "MESSAGGIO INTEGRO" oppure "MESSAGGIO CORROTTO". Se il vettore è integro, il programma visualizza anche il contenuto del messaggio. (suggerimento: rimuovere il bit di parità prima di procedere con la trasformazione di un numero in carattere).
   Esempio: il vettore dell'esercizio precedente [78, 207, 68, 201] rappresenta un messaggio integro di valore "NODI", il vettore [78, 207, **84**, 201] ha un bit alterato nel terzo byte e costituisce un messaggio corrotto.

4. [**Lcr1**] – Scrivere una funzione **Lcr** che riceva in ingresso un vettore di byte e applichi ai suoi elementi l'algoritmo LCR restituendo come risultato il corrispondente *Block Check Control* (BCC) .
   Scrivere inoltre un programma che: richieda l'inserimento di una stringa da tastiera e la converta in un vettore di byte **v1**; applichi agli elementi di **v1** l'algoritmo VCR e memorizzi il risultato in un vettore **v2**; applichi a **v2** l'algoritmo LCR per ricavare il **BCC**; concateni il contenuto di **v2** e **BCC** in un nuovo vettore **v3** e ne stampi il risultato.

Esempio: dalla stringa “NODI” si ricavano i seguenti ouput:

```
Codifica iniziale della stringa 'NODI':

   78       79       68       73
01001110 01001111 01000100 01001001

Codifica finale (algoritmo VCR + LCR)
   78      207       68      201    BCC[ 12]
01001110 11001111 01000100 11001001 BCC[00001100]
```

5. [**Lcr2**] – Scrivere un programma che simuli la ricezione di una sequenza binaria (rappresentata da un vettore di numeri) composta da valori codificati mediante VCR e da un BCC (vedi programma L**cr1**) verifichi l’integrità del messaggio contenuto. Il programma riutilizza le funzioni definite nei problemi precedenti per analizzare un vettore di numeri (costanti) e, in presenza di un errore, correggerlo (per semplicità si suppone che il numero massimo di bit errati sia pari a uno). Successivamente, il programma visualizza la frase “MESSAGGIO INTEGRO” oppure “ESEGUITA LA CORREZIONE DEL MESSAGGIO” seguita dal contenuto del messaggio.
   Esempio: il vettore dell’esercizio precedente [78, 207, 68, 201, 12] rappresenta un messaggio integro di valore “NODI”, il vettore [78, **203**, 68, 201, 12] ha un bit alterato nel secondo byte che, essendo individuabile grazie a LCR, può essere corretto.