RISOLTO

3. D
4. B
5. B
6. D
7. B-D
8. C-D
9. A
10. D

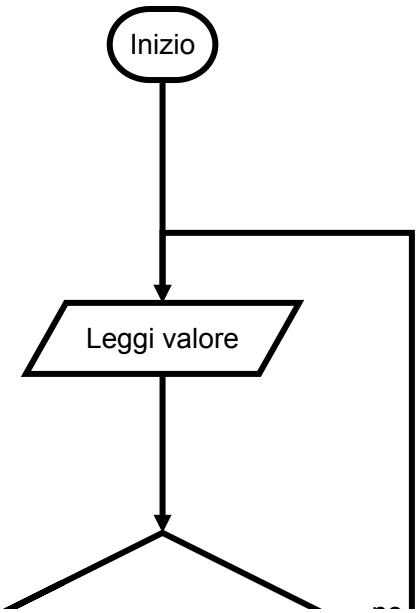
ES.1

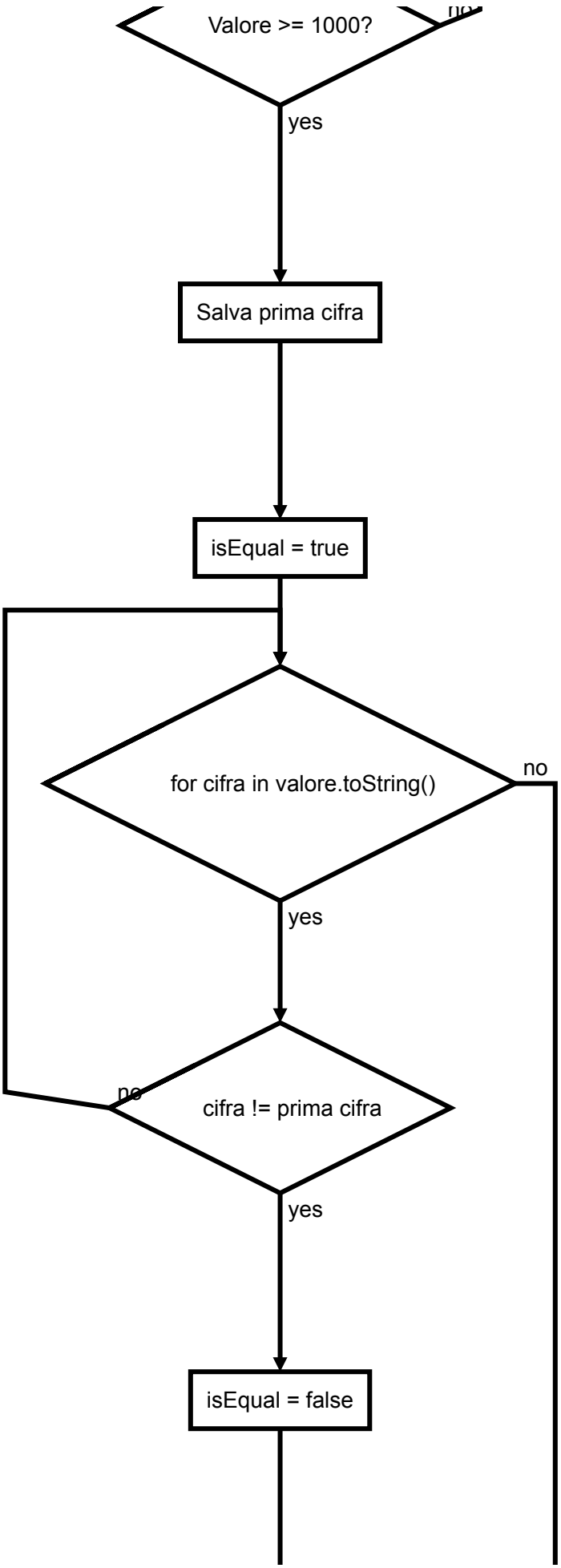
Siano dati i valori A=27 (8) e B=13(16) Li sI esprima entrambi in complemento a due sul numero minimo di bit Si calcoli A-B su tale numero di bit indicando riporti ed overfloW

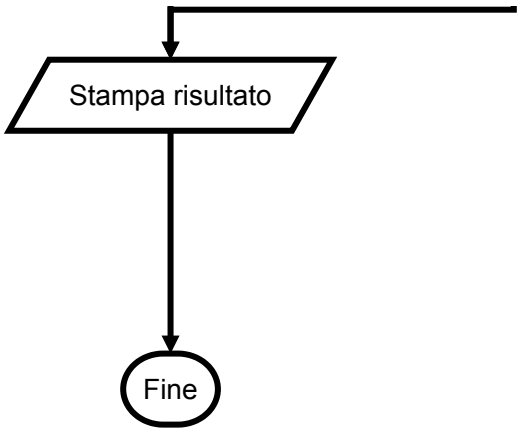
A 11011. B 1101

A−B=11011−11012 = 11010. No riporti o overflow.

ES. 2




Valore >= 1000?
yes
Salva prima cifra
isEqual = true
for cifra in valore.toString()
no
yes
cifra != prima cifra
no
yes
isEqual = false




```
def leggi_valore():
    valore = int(input("Inserisci un valore intero maggiore o uguale a 1000:
"))
    while valore < 1000:
        print("Il valore inserito non è maggiore o uguale a 1000. Riprova.")
        valore = int(input("Inserisci un valore intero maggiore o uguale a
1000: "))
    return valore

def verifica_ripetizione_cifra(valore):
    prima_cifra = str(valore)[0]
    isEqual = all(cifra == prima_cifra for cifra in str(valore))
    return isEqual

def stampa_risultato(is_ripetizione):
    if is_ripetizione:
        print("Si")
    else:
        print("No")

valore_inserito = leggi_valore()
is_ripetizione = verifica_ripetizione_cifra(valore_inserito)
stampa_risultato(is_ripetizione)
```

1. **Complessità Temporale:**
   - La funzione `leggi_valore` richiede al massimo un numero fisso di iterazioni $O(1)$.
   - La funzione `verifica_ripetizione_cifra` scorre tutte le cifre del valore una volta, quindi ha una complessità lineare $O(log_{10}N)$, dove N è il valore inserito.
2. **Complessità Spaziale:**
   - La funzione `leggi_valore` utilizza una quantità costante di memoria, indipendentemente dal valore di input $O(1)$.

- La funzione `verifica_ripetizione_cifra` utilizza una quantità costante di memoria aggiuntiva per variabili locali $O(1)$.

ES. 3

```python
def leggi_matrice():
    try:
        N = int(input("Inserisci la dimensione della matrice (N): "))
        if N <= 0:
            raise ValueError("Il valore deve essere un numero naturale maggiore
di 0.")

        matrice = []
        print("Inserisci i valori della matrice:")
        for _ in range(N):
            riga = [float(input()) for _ in range(N)]
            matrice.append(riga)

        return matrice
    except ValueError as e:
        print(f"Errore: {e}")
        return None

def calcola_determinante(matrice):
    if not matrice or len(matrice) == 0 or len(matrice) != len(matrice[0]):
        print("Matrice non valida.")
        return None

    N = len(matrice)

    if N == 1:
        return matrice[0][0]

    determinante = 0.0
    segno = 1  # Segno nel determinante di Laplace

    for j in range(N):
        sottomatrice = [riga[:j] + riga[j+1:] for riga in matrice[1:]]
        determinante += segno * matrice[0][j] *
calcola_determinante(sottomatrice)
        segno *= -1  # Alterna il segno per ogni termine nel determinante di
Laplace.

    return determinante

matrice = leggi_matrice()

if matrice:
    determinante = calcola_determinante(matrice)

    if determinante is not None:
        print(f"Il determinante della matrice è: {determinante}")
```