

# Olimpiadi di Informatica 2025/2026 - Selezione Scolastica

Cognome

Nome

Classe

Sezione

Data di nascita

Codice prova

399

Ciao! Ecco le informazioni essenziali sulla prova che stai per svolgere. Ti consigliamo di leggerle attentamente.

## Regole di base

Ti è permesso:

- avere a disposizione una calcolatrice (di qualunque tipo, comprese quelle grafiche);
- avere con te dei fogli bianchi (eventualmente a righe o quadretti);
- avere con te del materiale di cancelleria (penne, matite, gomma, etc.);
- andare in bagno (senza dispositivi elettronici o materiale cartaceo);
- comunicare con il docente sorvegliante in caso di problemi tecnici.

Non ti è permesso:

- comunicare con i tuoi compagni;
- comunicare con il docente sorvegliante sul contenuto della prova;
- diffondere il testo della prova, o parte di esso, prima delle ore 20 dell'ultimo giorno di prova (11 dicembre).

## Formato della prova

La prova contiene *10 problemi* da risolvere in *90 minuti*, ed è divisa in due parti:

- sei problemi di **pensiero logico-algoritmico**, e
- quattro problemi di interpretazione di **programmi** in pseudocodice o a blocchi.

In entrambe le parti i problemi sono **in ordine casuale**. Ogni problema comprende *due domande*, valutate separatamente. La prima è **sempre più semplice**, e può aiutare a rispondere alla seconda (che è solitamente difficile).

# Punteggio

Le domande possono essere a **risposta aperta** numerica; oppure a **scelta multipla** con cinque opzioni, di cui **solo una** è corretta. In ogni caso, il punteggio che puoi ottenere è:

- 5 punti per una risposta *corretta*;
- 1 punto per una risposta *non data*;
- 0 punti per una risposta *sbagliata*.

## Quesiti di programmazione e pseudocodice

I quesiti di programmazione presentano semplici programmi scritti in *pseudocodice*. Qui sotto puoi trovare un riassunto della sintassi dello pseudocodice. **Quest’anno, per la prima volta, i quesiti di programmazione verranno proposti anche come programma a blocchi:** potrai scegliere di guardare una versione, l’altra o entrambe, a seconda di come ti può essere più congeniale.

Pseudocodice	Descrizione	Esempio
■ Variabili e tipi		
<b>variable</b> <i>i</i> : <i>integer</i>	Dichiarazione della variabile di tipo intero chiamata <i>i</i>	
<b>variable</b> <i>arr</i> : <i>integer</i> []	Dichiarazione di una variabile di tipo array di interi chiamata <i>arr</i>	
<b>{var} ← {espr}</b>	Assegnamento del valore dell’espressione <b>{espr}</b> alla variabile <b>{var}</b>	<i>i</i> ← 1 <i>a</i> ← 3 × <i>i</i> + 5 <i>arr</i> ← [3, 5/ <i>a</i> , 2]
<i>arr</i> [ <b>{espr}</b> ]	Variabile corrispondente all’elemento dell’array <i>arr</i> di indice <b>{espr}</b>	<i>a</i> ← <i>arr</i> [3] + 1 <i>arr</i> [ <i>x</i> + 1] ← 2
<b>(a, b) ← (b, a)</b>	Scambio del valore delle variabili <i>a</i> e <i>b</i>	
■ Operatori		
<b>+</b> , <b>−</b> , <b>×</b> , <b>/</b> , <b>mod</b>	Aritmetica: addizione, sottrazione (o negazione), moltiplicazione, divisione intera, resto della divisione intera (modulo)	<i>a</i> + <i>b</i> − <i>a</i> <i>i</i> ← <i>a</i> mod 10
<b>==</b> , <b>≠</b> , <b>&lt;</b> , <b>≤</b> , <b>&gt;</b> , <b>≥</b>	Confronto: uguale, diverso, minore, minore o uguale, maggiore, maggiore o uguale	<i>a</i> == 7 2 × ( <i>x</i> + 1) ≤ <i>y</i>
<b>and</b> , <b>or</b> , <b>not</b>	Operatori logici: e, o, non	<i>a</i> > <i>b</i> and <i>b</i> ≠ −1 not ( <i>a</i> > 2 or <i>a</i> == 0)
■ Strutture di controllo		
<b>if</b> {condizione} <b>then</b> {corpo if} <b>else</b> {corpo else} <b>end if</b>	Struttura condizionale <b>if ... else</b> : se {condizione} è vera viene eseguito {corpo if}, altrimenti viene eseguito {corpo else}. La parte <b>else</b> può essere omessa	<b>if</b> <i>n</i> mod 2 == 0 <b>then</b> <i>i</i> ← 0 <b>else</b> <i>i</i> ← <i>n</i> − 1 <b>end if</b>
<b>while</b> {condizione} <b>do</b> {corpo} <b>end while</b>	Ciclo <b>while</b> : il blocco {corpo} viene ripetuto fintanto che {condizione} è vera	<b>while</b> <i>i</i> < <i>n</i> <b>do</b> <i>sum</i> ← <i>sum</i> + <i>i</i> <i>i</i> ← <i>i</i> + 7 <b>end while</b>
<b>for</b> {indice} <b>in</b> {intervallo} <b>do</b> {corpo} <b>end for</b>	Ciclo <b>for</b> : il blocco {corpo} viene eseguito mentre la variabile {indice} itera sui valori in {intervallo}, specificato come [ <i>a</i> ... <i>b</i> ], che significa “tutti i numeri da <i>a</i> (incluso) fino a <i>b</i> (escluso)”	<b>for</b> <i>i</i> <b>in</b> [0 ... <i>n</i> ) <b>do</b> <i>arr</i> [ <i>i</i> ] ← −1 <b>end for</b> (assegna −1 a tutti gli elementi di un array <i>arr</i> di lunghezza <i>n</i> )
■ Funzioni		
<b>function</b> <b>fun</b> ( <i>var1</i> : <i>tipo1</i> , <i>var2</i> : <i>tipo2</i> , ...) → <i>ritorno</i> {corpo} <b>end function</b>	Funzione con parametri <i>var1</i> , <i>var2</i> , etc. Il tipo di ritorno → <i>ritorno</i> può essere omesso. Il valore viene restituito tramite la parola chiave <b>return</b>	<b>function</b> <b>add</b> ( <i>a</i> : <i>integer</i> , <i>b</i> : <i>integer</i> ) → <i>integer</i> <b>return</b> <i>a</i> + <i>b</i> <b>end function</b>
<b>fun</b> () <b>fun</b> ( <i>arg1</i> , <i>arg2</i> , ...)	Chiamata alla funzione <b>fun</b> (rispettivamente senza argomenti e con argomenti). La funzione <b>output</b> stampa il valore di una variabile oppure una stringa fissata. Le funzioni <b>min</b> e <b>max</b> restituiscono risp. il minimo e il massimo di due interi	<b>return</b> <b>add</b> ( <i>a</i> , <i>b</i> ) <i>m</i> ← <b>very_big_integer</b> () <b>output</b> ( <i>x</i> ) <b>output</b> ("string") <b>min</b> ( <i>a</i> , <i>b</i> ) <b>max</b> ( <i>a</i> , <i>b</i> )



### Domanda 3.1

Nel nuovo centro commerciale *WallMars* ci sono 128 porte automatiche, inizialmente tutte disattivate, e ognuna collegata ad un interruttore per attivarne il funzionamento. Da una soffiata anonima, hai scoperto che **esattamente uno** degli interruttori è rotto, ma non sai quale e devi scoprirlo!



Ogni minuto puoi fare esattamente una di queste 2 azioni:

1. azionare un interruttore: se è quello rotto non succede nulla, altrimenti cambia lo stato della porta (da disattiva a attiva oppure da attiva a disattiva, ma senza scoprire se alla fine è attiva o disattiva);
2. controllare se una data porta è attiva o disattiva.

Di quanti minuti hai bisogno al minimo per essere sicuro di capire qual è l'interruttore rotto, anche se sei sfortunato?

- ☐ A) 256    ☐ B) 254    ☐ C) 127  
☐ D) 192    ☐ E) 128

### Domanda 3.2

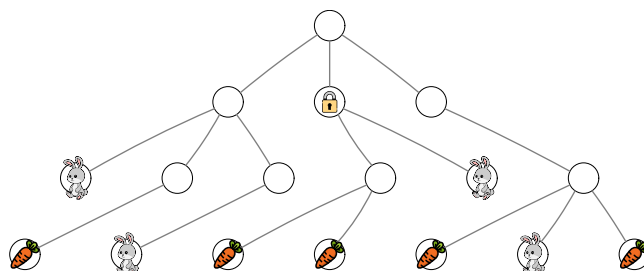
Dopo averci pensato meglio, ti sei accorto che c'è una terza azione che puoi fare, sempre nel tempo di un minuto:

3. misurare il consumo elettrico totale di *WallMars*, capendo così esattamente quante porte sono attive in quel momento.

Con quest'altra possibilità, e le due precedenti, di quanti minuti hai bisogno per capire qual è l'interruttore rotto?

### Domanda 4.1

Hai appena aperto un allevamento di conigli. L'allevamento comprende diverse aree, collegate da percorsi secondo questa struttura ad albero:

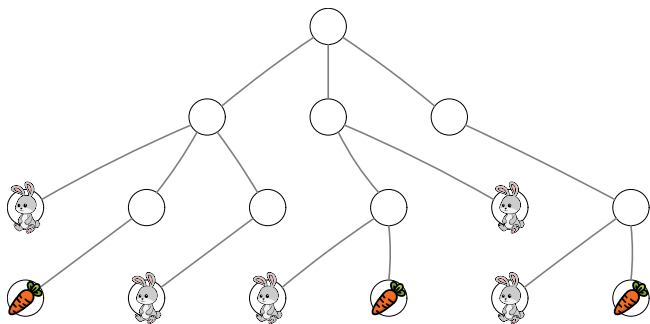


Alcune delle aree dell'allevamento contengono conigli, altre contengono dei campi di carote. I percorsi interni invece possono essere bloccati da una recinzione, indicata nello schema da un lucchetto, che i conigli non possono attraversare. Nella configurazione sopra, quante sono le diverse **coppie** (coniglio, carota) per cui il coniglio può raggiungere la carota?

- ☐ A) 6    ☐ B) 12    ☐ C) 3  
☐ D) 9    ☐ E) 20

Domanda 4.2

L'ora del pranzo è terminata, e ora vuoi evitare che i conigli possano raggiungere ancora le carote. Se l'allevamento segue questo schema:

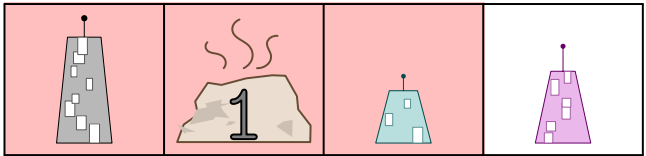


Quanti percorsi interni (cerchi bianchi) devi bloccare al minimo di modo che nessun coniglio possa raggiungere nessuna carota?

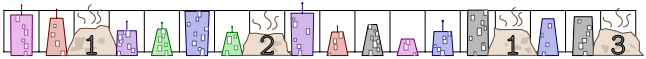
- ☐ A) 3
- ☐ B) 2
- ☐ C) 4
- ☐ D) 1
- ☐ E) 5

Domanda 5.1

La città di *Scart City* è famosa per le sue numerose discariche. Ogni discarica ha un *livello di odorosità*  $k$ , che vuol dire che se ne sente l'odore in tutte le case fino ad una distanza di  $k$ . Per esempio, in questa configurazione solo la casa più a destra è abbastanza lontana dalla discarica:



Vuoi acquistare alcune case nel *quartiere stretto*, che è una linea di 18 isolati come in figura:

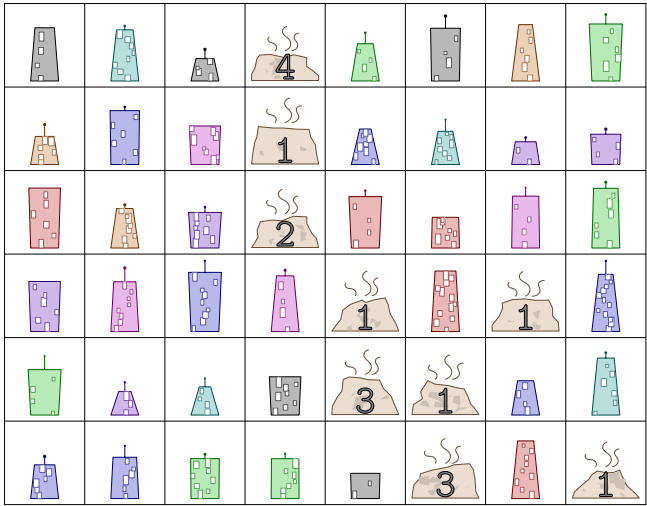


Quante sono le case del quartiere stretto da cui non si sente l'odore di nessuna discarica?

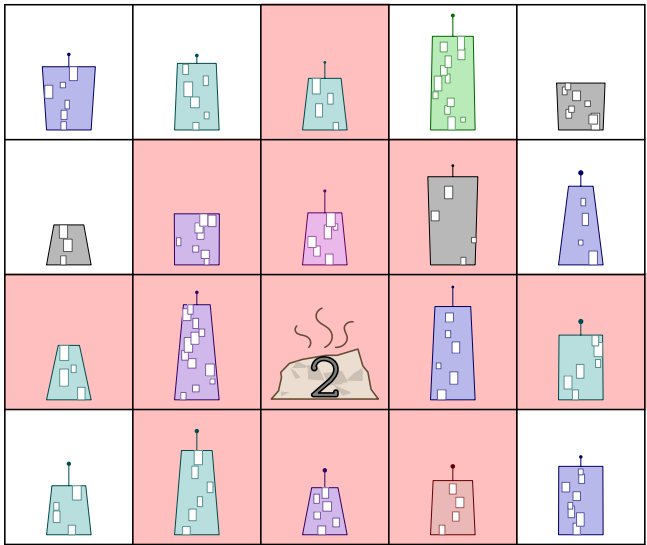
- ☐ A) 7
- ☐ B) 5
- ☐ C) 2
- ☐ D) 1
- ☐ E) 9

Domanda 5.2

Sei anche interessato alle case nel *quartiere quadro*, che è composto da  $8 \times 6$  isolati come in figura:



Attenzione che l'odore si propaga a tutti gli isolati direttamente adiacenti, ma **non in diagonale**. Per esempio, in questa configurazione ci sono 8 case abbastanza lontane dalla discarica:

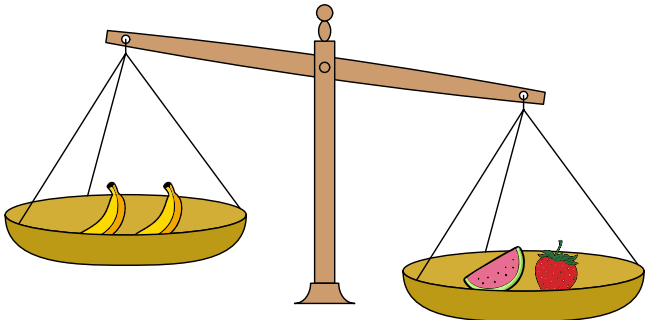
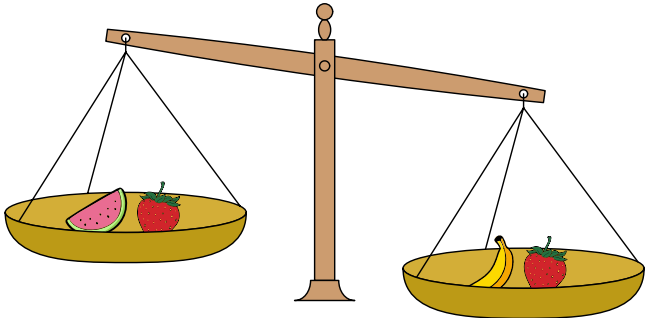


Quante sono le case del quartiere quadro da cui non si sente l'odore di nessuna discarica?

- ☐ A) 8
- ☐ B) 1
- ☐ C) 7
- ☐ D) 2
- ☐ E) 9

Domanda 6.1

Oggi hai comprato alcuni frutti e poi hai provato a pesarli sulla tua bilancia, come vedi qui:

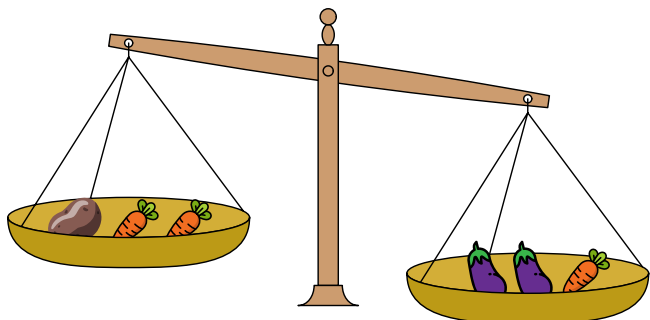


In tutte e due le prove le bilance sono squilibrate: il peso totale degli oggetti a sinistra è meno del peso totale degli oggetti a destra. Sai che i frutti di tipo uguale pesano tutti uguale. Qual è l'ordine di peso dei frutti (dal più leggero al più pesante)?

- ☐ A) leggero: medio: pesante:
- ☐ B) leggero: medio: pesante:
- ☐ C) leggero: medio: pesante:
- ☐ D) leggero: medio: pesante:
- ☐ E) leggero: medio: pesante:

## Domanda 6.2

Visto che hai anche comprato alcune verdure (patate, carote e melanzane), provi a pesare anche quelle sulla bilancia:



Da questi risultati, cosa puoi dedurre sul peso di queste tre verdure?

- ☐ A) la patata è la più leggera, ma non si può sapere niente sulle altre due verdure
- ☐ B) la carota è la più leggera, la melanzana è intermedia e la patata è la più pesante
- ☐ C) non si può dedurre niente sui pesi delle tre verdure
- ☐ D) la melanzana è la più leggera, ma non si può sapere niente sulle altre due verdure
- ☐ E) la patata è la più leggera, la melanzana è intermedia e la carota è la più pesante

## Sezione 2: Esercizi di programmazione

### Domanda 7.1

Un *array*  $a$  di lunghezza  $n$  è una sequenza di numeri interi indicizzati da zero, che indichiamo con  $a[0], a[1], \dots, a[n-1]$ .

Durante una pausa dallo studio del greco antico hai scoperto la funzione **ribalta**! Questa funzione dato un *array*, un indice di **inizio** e una **lunghezza**, modifica l'*array* invertendo l'ordine dei suoi primi **lunghezza** elementi a partire da **array[inizio]**:

```
function ribalta(array: integer[], inizio: integer, lunghezza: integer)
    fine ← inizio + lunghezza - 1
    for i in [0 ... lunghezza / 2) do
        (array[inizio + i], array[fine - i])
        ← (array[fine - i], array[inizio + i])
    end for
end function
```

per fare **ribalta** con **array** e **inizio** e **lunghezza** :

imposta **fine** a  $(\text{inizio} + \text{lunghezza}) - 1$

conta con **i** da 0 a  $(\text{lunghezza} \div 2) - 1$  per 1 :

scambia valore di **array**  $(\text{inizio} + i)$  e **array**  $(\text{fine} - i)$

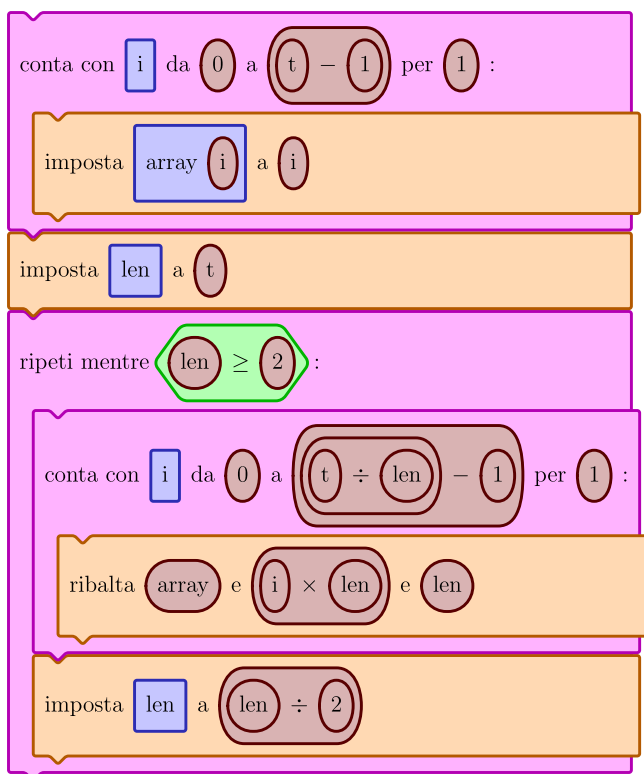
Ora vuoi usare questa funzione per generare un codice segreto! Dato un *array* di lunghezza totale  $t = 8$ , esegui questo procedimento:

```
for i in [0 ... t) do
    array[i] ← i
end for
len ← t
while len ≥ 2 do
```

```

for i in [0 ... t / len) do
    ribalta(array, i × len, len)
end for
len ← len / 2
end while

```



Alla fine del procedimento, che numero sarà contenuto in **array[3]**?

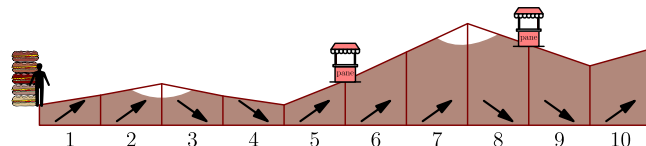
- ☐ A) 5    ☐ B) 6    ☐ C) 3  
☐ D) 7    ☐ E) 2

## Domanda 7.2

Non essendo soddisfatto di un codice così corto, decidi di ripetere il procedimento con un **array** di lunghezza totale  $t = 1024$ . Alla fine del procedimento, che numero sarà contenuto in **array[29]**?

## Domanda 8.1

Vuoi andare a fare una passeggiata in montagna, portandoti dietro 5 panini:



La passeggiata prevede 10 tratti di percorso e passa per alcuni chioschi del pane. In ogni tratto, vuoi seguire questo procedimento:

```

if chiosco() then
    compra_panini(2)
    if salita() then
        mangia_panini(2)
    end if
else
    if salita() then
        mangia_panini(1)
    end if
end if
cammina()

```





Con quanti panini rimarrai alla fine del percorso?

- ☐ A) non ti rimarrà nessun panino
- ☐ B) 1    ☐ C) 3    ☐ D) 5
- ☐ E) 2

## Domanda 8.2

Una volta arrivato, ti sei accorto di aver dimenticato i panini! C'è solo una soluzione: sperare che aprano altri chioschi nel frattempo. Quanti chioschi dovrebbero aggiungersi **al minimo**, perché tu possa riuscire ad arrivare fino in fondo al percorso rispettando il procedimento, e partendo **senza panini**?

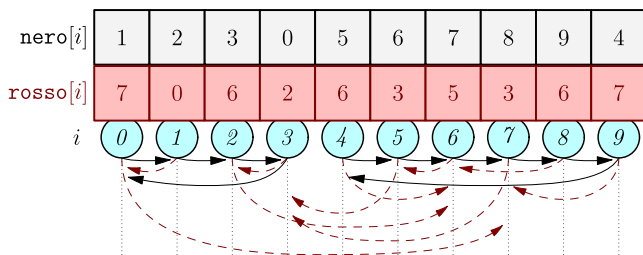
- ☐ A) 4    ☐ B) 1    ☐ C) 3
- ☐ D) non serve aggiungere nessun chiosco
- ☐ E) 2

## Domanda 9.1

Un array  $a$  di lunghezza  $n$  è una sequenza di numeri interi indicizzati da zero, che indichiamo con  $a[0], a[1], \dots, a[n-1]$ .

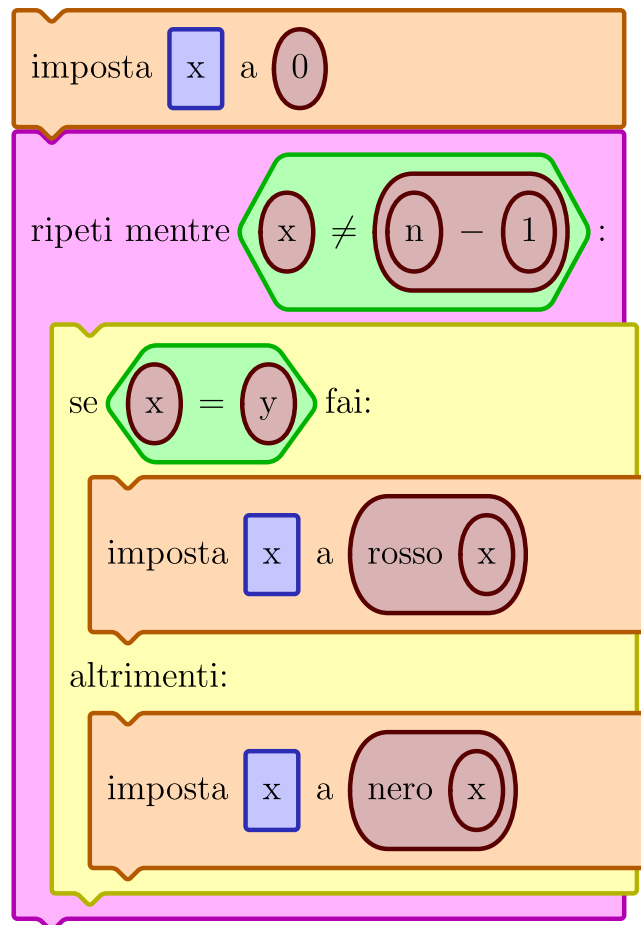
Durante la tua ultima passeggiata in montagna, hai trovato due array lungo il percorso: **nero** e **rosso**, entrambi della stessa lunghezza  $n = 10$ .

Rappresentando **nero** $[x]$  con una freccia nera da  $x$  a **nero** $[x]$ , e **rosso** $[x]$  con una freccia rossa tratteggiata da  $x$  a **rosso** $[x]$ , gli array sono:



Vorresti ora passeggiare lungo i due array seguendo questo procedimento, dato un certo valore  $y$  che devi scegliere:

```
x ← 0
while x ≠ n - 1 do
  if x == y then
    x ← rosso[x]
  else
    x ← nero[x]
  end if
end while
```



Quanti diversi valori di  $y$  tra 0 e 9 potresti scegliere di modo che la passeggiata termini, senza andare avanti all'infinito?

- ☐ A) 5
- ☐ B) non ce ne sono, la passeggiata va sempre avanti all'infinito
- ☐ C) 4    ☐ D) 1    ☐ E) 2

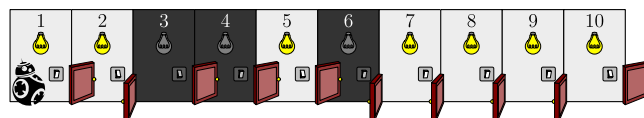
### Domanda 9.2

Oltre a non voler passeggiare all'infinito, vorresti anche assicurarti di visitare **tutte le posizioni** tra 0 e 9. Per fortuna, ti sei accorto che potresti anche modificare un valore dell'array **ROSSO**! Per quale coppia di valori **y** e **rosso[y]** riusciresti a visitare tutti i numeri?

Indica la risposta come **AB**, dove **A** è il valore di **y** e **B** è il nuovo valore da assegnare a **rosso[y]**.

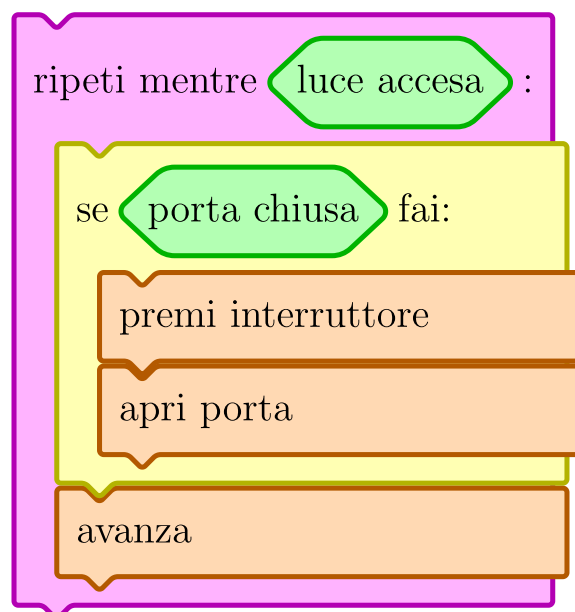
### Domanda 10.1

Hai comprato un nuovo robot tuttofare, e vuoi fargli pulire casa! Il tuo appartamento è composto da 10 stanze separate da porte, ciascuna con un interruttore che regola la luce della stanza successiva. Inizialmente, il robot si trova all'inizio del tuo appartamento, e le luci e porte sono in questo stato:



Hai programmato il robot per seguire questo procedimento:

```
while luce_accessa() do
    if porta_chiusa() then
        premi_interruttore()
        apri_porta()
    end if
    avanza()
end while
```

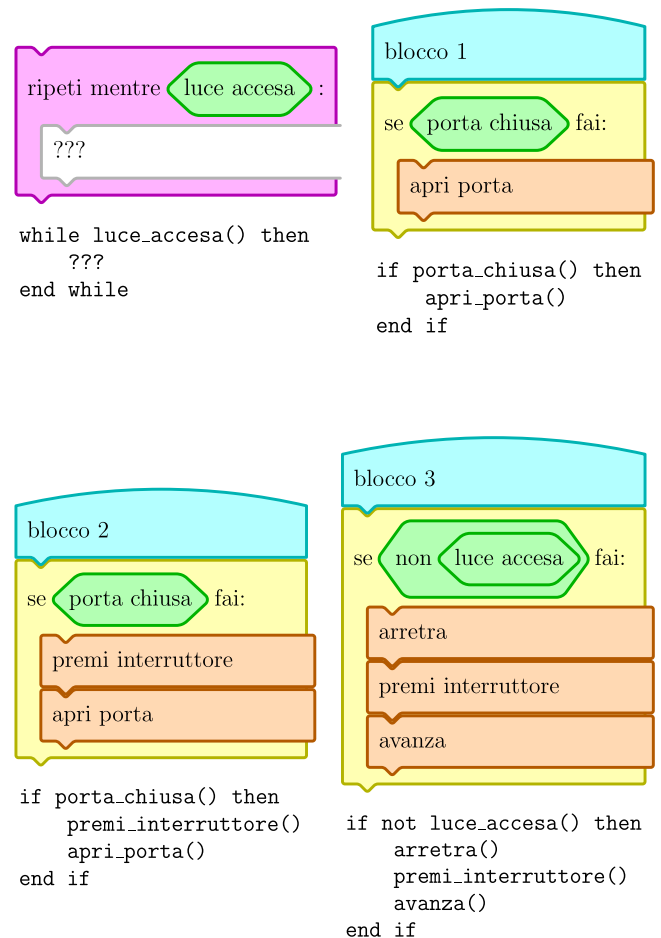


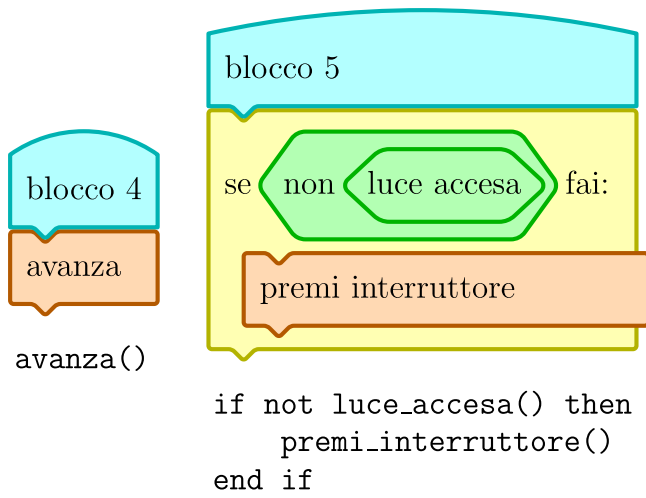
Dove si troverà il robot alla fine del procedimento?

- ☐ A) nella stanza 8      ☐ B) nella stanza 4
- ☐ C) nella stanza 3
- ☐ D) il robot esce dall'appartamento
- ☐ E) nella stanza 9

## Domanda 10.2

Non sei molto convinto del procedimento che hai programmato prima, allora hai deciso di modificarlo di modo che **qualunque** sia lo stato delle porte e delle luci, il robot arrivi sempre fino alla fine dell'appartamento, **a patto che la luce nella stanza iniziale sia accesa**. Purtroppo però il robot ha una programmazione molto vincolata! Devi per forza tenere il ciclo "ripeti mentre" più esterno, mentre al suo interno puoi mettere in sequenza i blocchi di istruzioni che preferisci, scegliendo tra questi 5 possibili:





Per riportare la programmazione al robot, devi quindi scrivere una sequenza di numeri da 1 a 5 (anche ripetuti), che corrisponde ai blocchi di istruzioni da eseguire nel ciclo. Per esempio, il programma riportato nella domanda precedente corrisponde al programma **24**, ma purtroppo non sempre consente al robot di arrivare in fondo all'appartamento. Quale sarebbe invece un programma corretto? Tieni conto che il robot non può avanzare se la porta è chiusa.