Olimpiadi di Informatica 2025/2026 - Selezione Scolastica

Cognome

Nome

Classe

Sezione

Data di nascita

Codice prova

387

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		
`variable i: integer`	Dichiarazione della variabile di tipo intero chiamata `i`	
`variable arr: integer[]`	Dichiarazione di una variabile di tipo array di interi chiamata `arr`	

`{var} ← {espr}`	Assegnamento del valore dell'espressione `{espr}` alla variabile `{var}`	`i ← 1`
		`a ← 3 × i + 5`
		`arr ← [3, 5/a, 2]`

`arr[{espr}]`	Variabile corrispondente all'elemento dell'array `arr` di indice `{espr}`	`a ← arr[3] + 1` `arr[x + 1] ← 2`
`(a, b) ← (b, a)`	Scambio del valore delle variabili `a` e `b`	
■ Operatori		

`+, −, ×, /, mod`	Aritmetica: addizione, sottrazione (o negazione), moltiplicazione, divisione intera, resto della divisione intera (modulo)	`a + b`
		`−a`
		`i ← a mod 10`

`==, ≠, <, ≤, >, ≥`	Confronto: uguale, diverso, minore, minore o uguale, maggiore, maggiore o uguale	`a == 7` `2 × (x + 1) ≤ y`
`and, or, not`	Operatori logici: e, o, non	`a > b and b ≠ -1` `not (a > 2 or a == 0)`
■ Strutture di controllo		

`if {condizione} then`
`  {corpo if}`
`else`
`  {corpo else}`
`end if`
	Struttura condizionale `if ... else`: se `{condizione}` è vera viene eseguito `{corpo if}`, altrimenti viene eseguito `{corpo else}`.
	La parte `else` può essere omessa
		`if n mod 2 == 0 then`
		`  i ← 0`
		`else`
		`  i ← n − 1`
		`end if`

`while {condizione} do`
`  {corpo}`
`end while`
	Ciclo `while`: il blocco `{corpo}` viene ripetuto fintanto che `{condizione}` è vera	`while i < n do`
		`  sum ← sum + i`
		`  i ← i + 7`
		`end while`

`for {indice} in {intervallo} do`
`  {corpo}`
`end for`
	Ciclo `for`: il blocco `{corpo}` viene eseguito mentre la variabile `{indice}` itera sui valori in `{intervallo}`, specificato come `[a ... b)`, che significa "tutti i numeri da a (incluso) fino a b (escluso)"	`for i in [0 ... n) do`
		`  arr[i] ← -1`
		`end for`
		(assegna `-1` a tutti gli elementi di un array `arr` di lunghezza `n`)

■ Funzioni		

`function`
`fun(var1: tipo1, var2: tipo2, ...)`
`→ ritorno`
`  {corpo}`
`end function`
	Funzione con parametri `var1`, `var2`, etc.
	Il tipo di ritorno `→ ritorno` può essere omesso.
	Il valore viene restituito tramite la parola chiave `return`
		`function`
		`add(a: integer, b: integer) →`
		`integer`
		`  return a + b`
		`end function`

`fun()`
`fun(arg1, arg2, ...)`
	Chiamata alla funzione `fun` (rispettivamente senza argomenti e con argomenti).
	La funzione `output` stampa il valore di una variabile oppure una stringa fissata.
	Le funzioni `min` e `max` restituiscono risp. il minimo e il massimo di due interi
		`return add(a, b)`
		`m ← very_big_integer()`
		`output(x)`
		`output("string")`
		`min(a, b)`
		`max(a, b)`

Sezione 1: Esercizi logico-algoritmici

Domanda 1.1

Nel nuovo centro commerciale *WallMars* ci sono 1024 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) 2046 B) 1024 C) 1536 D) 1023 E) 2048

Domanda 1.2

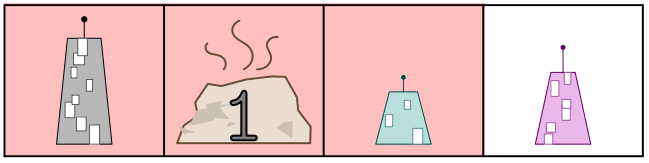
Dopo averci pensato meglio, ti sei accorto che c'è una terza azione che puoi fare, sempre nel tempo di un minuto:

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

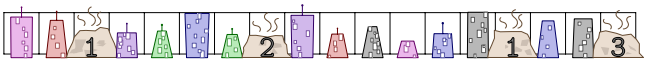
Con quest'altra possibilità, e le due precedenti, di quanti minuti hai bisogno per capire qual è l'interruttore rotto?

Domanda 2.1

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:

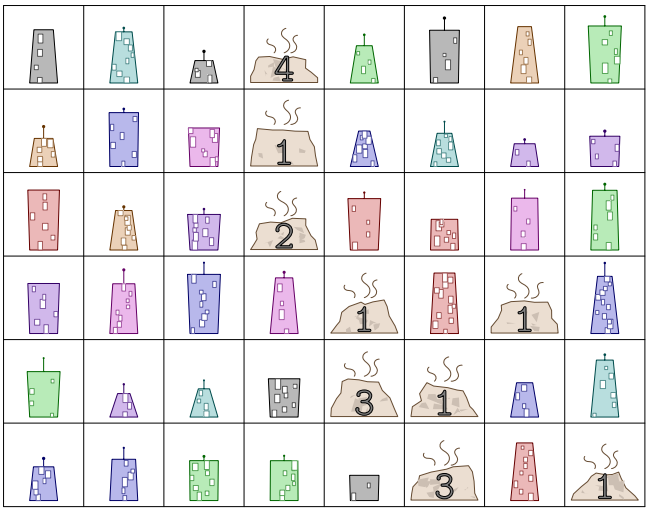


Quante sono le case del quartiere stretto da cui non si sente l'odore di nessuna discarica?

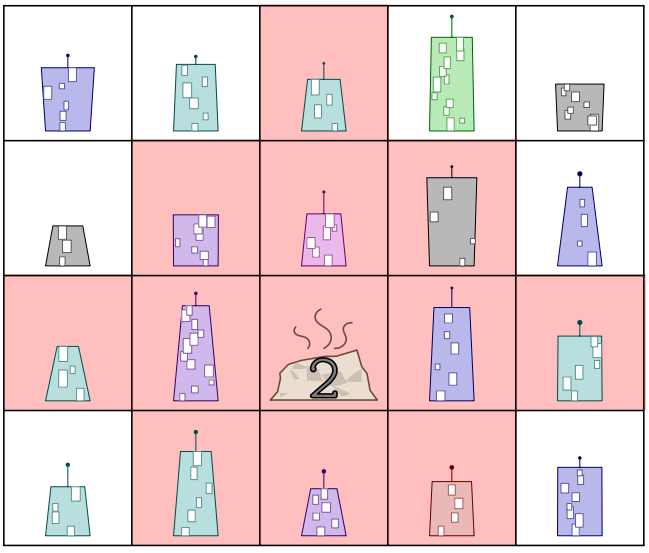
A) 5 B) 2 C) 7 D) 1 E) 9

Domanda 2.2

Sei anche interessato alle case nel *quartiere quadro*, che è composto da 8×6 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:

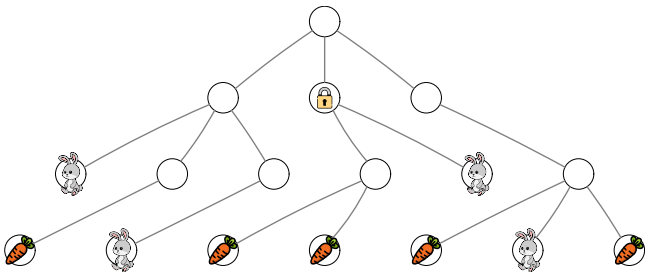


Quante sono le case del quartiere quadro da cui non si sente l'odore di nessuna discarica?

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

Domanda 3.1

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

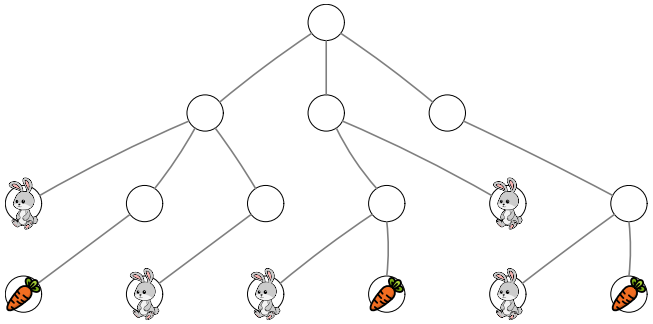


Alcune delle aree dell'allevamento contengono conigli, altre contengono dei campi di carote. I percorsi interni invece possono essere bloccati da una recinzione, indicata nello schema da un lucchetto, che i conigli non possono attraversare. Nella configurazione sopra, quante sono le diverse **coppie**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

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

Domanda 3.2

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

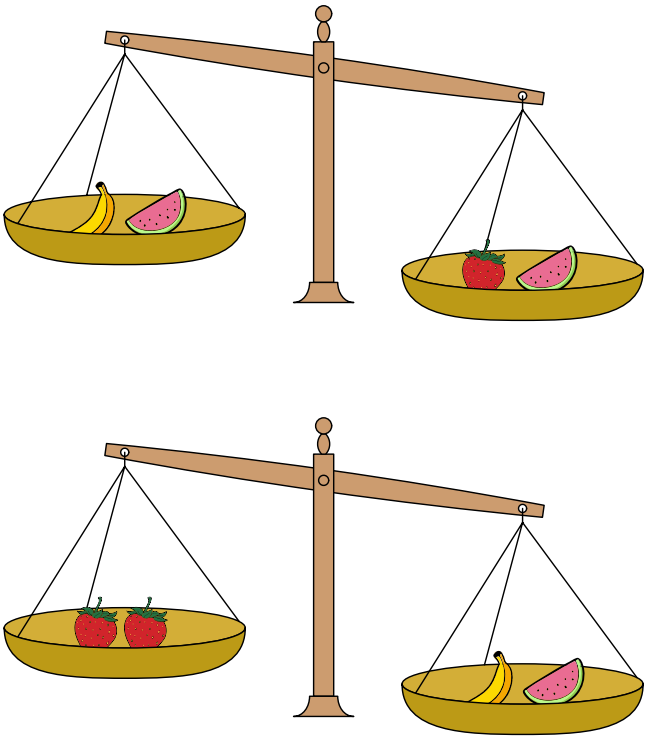


Quanti percorsi interni (cerchi bianchi) devi bloccare al minimo di modo che nessun coniglio possa raggiungere nessuna carota?

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

Domanda 4.1

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

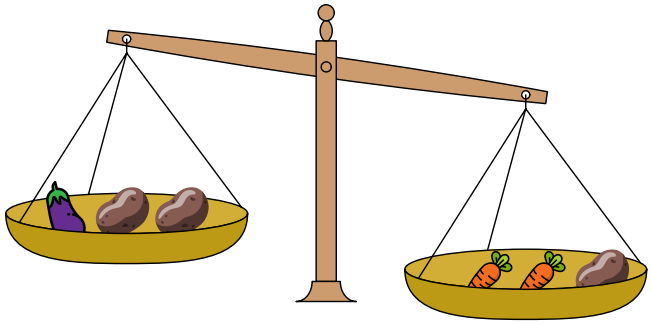


In tutte e due le prove le bilance sono squilibrate: il peso totale degli oggetti a sinistra è meno del peso totale degli oggetti a destra. Sai che i frutti di tipo uguale pesano tutti uguale. Qual è l'ordine di peso dei frutti (dal più leggero al più pesante)?

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

Domanda 4.2

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 melanzana è la più leggera, la carota è intermedia e la patata è la più pesante

B) la melanzana è la più leggera, ma non si può sapere niente sulle altre due verdure

C) la patata è la più leggera, la carota è intermedia e la melanzana è la più pesante

D) la carota è la più leggera, ma non si può sapere niente sulle altre due verdure

E) non si può dedurre niente sui pesi delle tre verdure

Domanda 5.1

Per la verifica di *greco antico* devi studiare 5 pile di libri di varie altezze. Mentre li osservi, raccogliendo le forze per iniziare a leggerli, sei un po' infastidito dal fatto che le pile hanno altezze diverse. Quanti libri dovresti **aggiungere** al minimo per portare tutte le pile alla stessa altezza?

A) 8 B) 2 C) 5 D) 7 E) 6

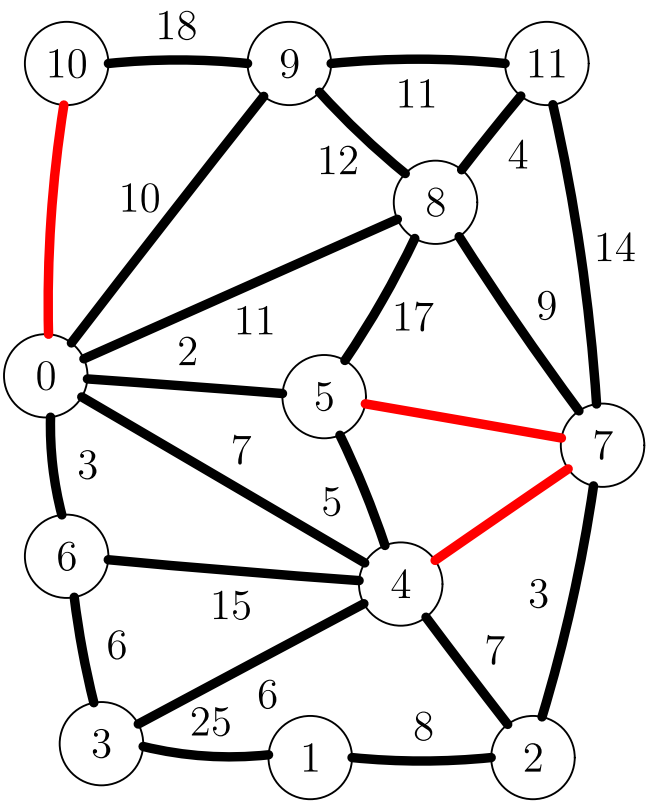
Domanda 5.2

Osservando ancora meglio le pile, hai pensato che non solo potresti **aggiungere** nuovi libri prendendoli dalla tua riserva, ma potresti anche **spostare** i libri che ci sono da una pila all'altra. Quanti libri dovresti **spostare o aggiungere** al minimo, per portare tutte le pile alla stessa altezza?

A) 2 B) 7 C) 6 D) 3 E) 4

Domanda 6.1

A Olinfolandia ci sono 12 città collegate da strade. Solo alcune di queste strade, tuttavia, sono ad alta velocità (in rosso):



Qual è il numero minimo di strade normali (nere) che dovrebbero essere potenziate ad alta velocità (colorate di rosso), di modo che sia possibile raggiungere qualsiasi città partendo da qualsiasi altra città percorrendo solo strade ad alta velocità?

- A) 6
- B) 5
- C) 9
- D) 8
- E) 13

Domanda 6.2

Ora vorresti sapere anche quanto ti *costerebbe* al minimo collegare tutta Olinfolandia all'alta velocità, sapendo che ogni strada normale ha un diverso costo per essere potenziata (il numero riportato sopra la strada). Qual è il costo totale minimo?

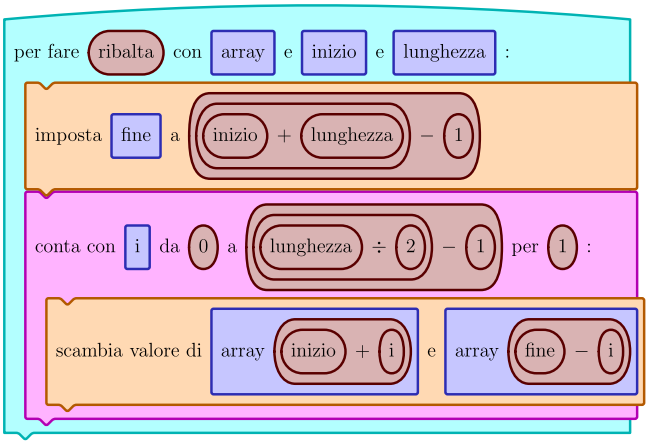
Sezione 2: Esercizi di programmazione

Domanda 7.1

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

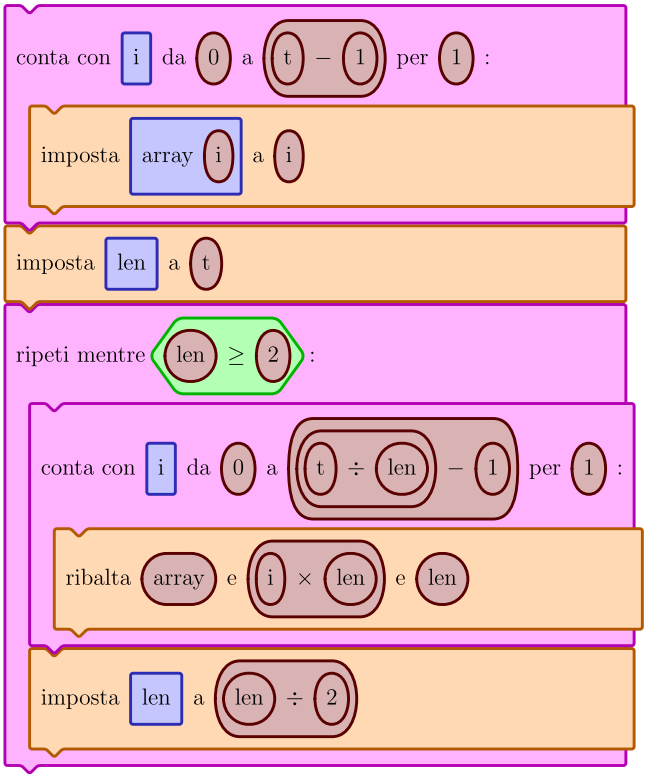
Durante una pausa dallo studio del greco antico hai scoperto la funzione `ribalta`! Questa funzione dato un `array`, un indice di `inizio` e una `lunghezza`, modifica l'array invertendo l'ordine dei suoi primi `lunghezza` elementi a partire da `array[inizio]`:

```
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
```



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[4]`?

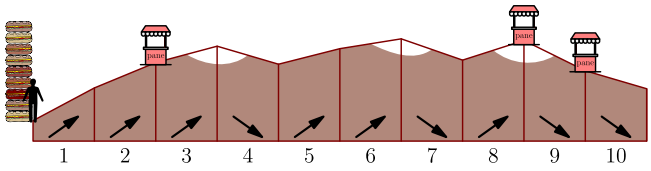
A) 3 B) 6 C) 5 D) 1 E) 7

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[252]`?

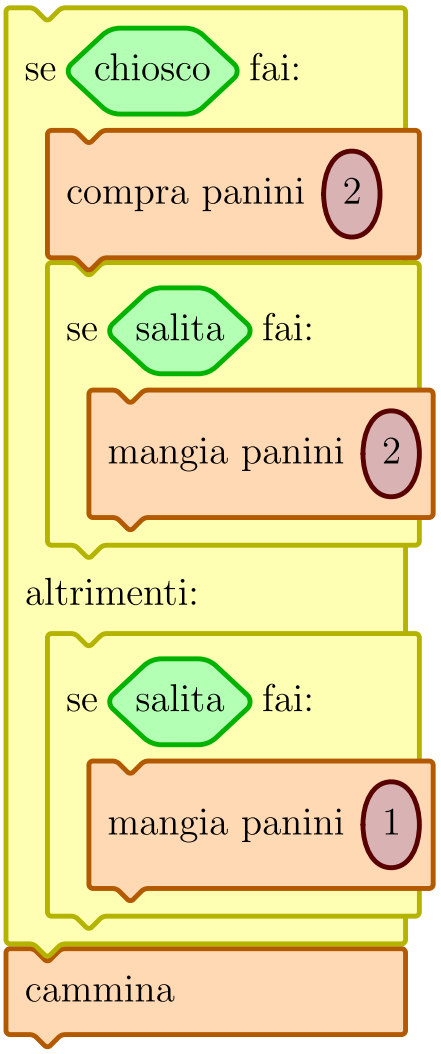
Domanda 8.1

Vuoi andare a fare una passeggiata in montagna, portandoti dietro 9 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) 9 B) 7 C) 10

D) 8

E) non ti rimarrà nessun panino

Domanda 8.2

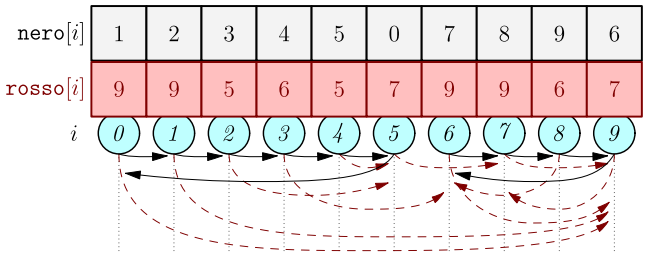
Una volta arrivato, ti sei accorto di aver dimenticato i panini! C'è solo una soluzione: sperare che aprano altri chioschi nel frattempo. Quanti chioschi dovrebbero aggiungersi **al minimo**, perché tu possa riuscire ad arrivare fino in fondo al percorso rispettando il procedimento, e partendo **senza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

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

Domanda 9.1

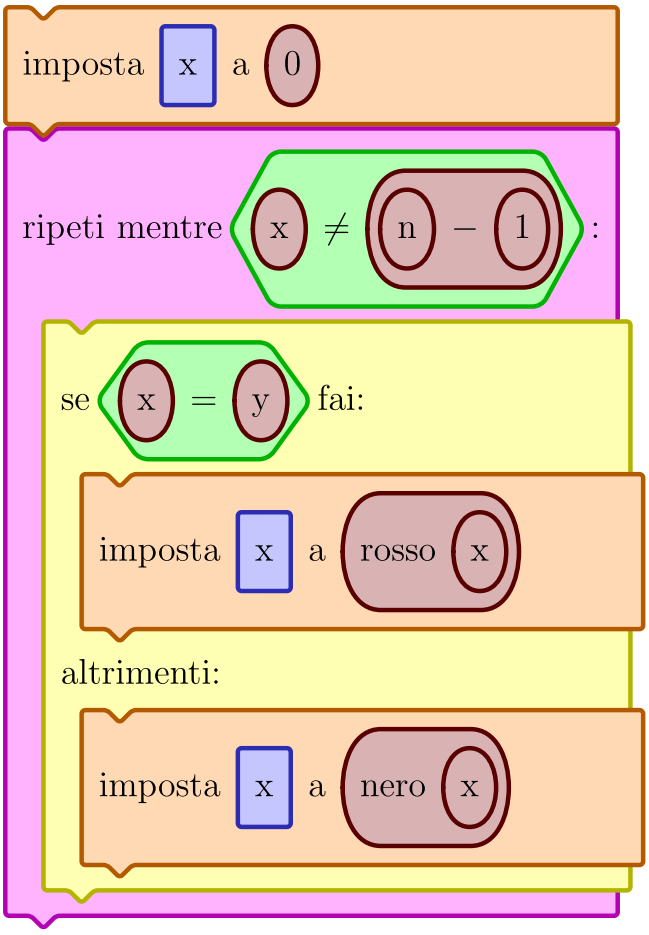
Un *array* a di lunghezza n è una sequenza di numeri interi indicizzati da zero, che indichiamo con $a[0], a[1], \ldots a[n-1]$.

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) 4 B) 1

C) non ce ne sono, la passeggiata va sempre avanti all'infinito

D) 3 E) 5

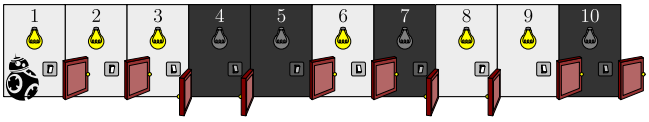
Domanda 9.2

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]`.

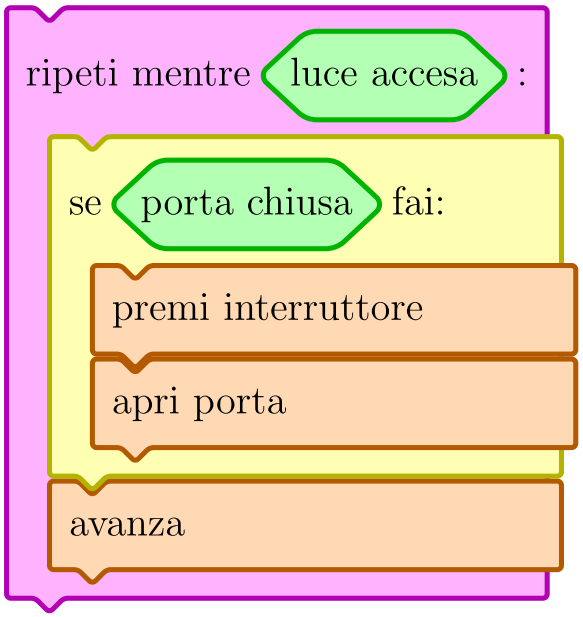
Domanda 10.1

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



Dove si troverà il robot alla fine del procedimento?

A) il robot esce dall'appartamento

B) nella stanza 8

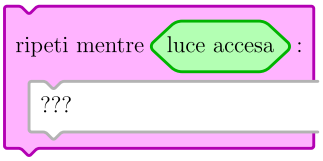
C) nella stanza 7

D) nella stanza 4

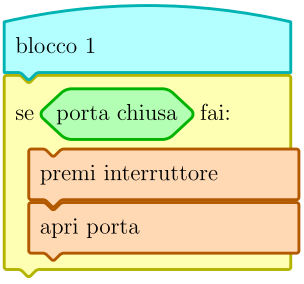
E) nella stanza 10

Domanda 10.2

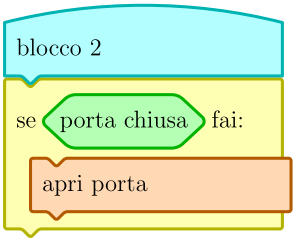
Non sei molto convinto del procedimento che hai programmato prima, allora hai deciso di modificarlo di modo che **qualunque** sia lo stato delle porte e delle luci, il robot arrivi sempre fino alla fine dell'appartamento, **a patto che la luce nella stanza iniziale sia accesa**. Purtroppo però il robot 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:



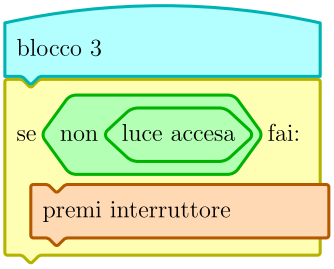

```
while luce_accesa() then
    ???
end while
```




```
if porta_chiusa() then
    premi_interruttore()
    apri_porta()
end if
```




```
if porta_chiusa() then
    apri_porta()
end if
```




```
if not luce_accesa() then
    premi_interruttore()
end if
```


```
avanza()
```


```
if not luce_accesa() then
    arretra()
    premi_interruttore()
    avanza()
end if
```

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 **14**, 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.