Esercizio. Si scriva in Python una funzione booleana che ricevute come parametro due parole di uguale lunghezza, verifichi se sono l'una anagramma dell'altra. Due parole sono una l'anagramma dell'altra se contengono le stesse lettere ma in ordine diverso. *Esempio*: La parola ***locandiera*** è un anagramma della parola ***calendario***.
Se le due parole non sono una l'anagramma dell'altra, si consideri la prima inserita da input e se ne costruisca un anagramma permutando in modo casuale le lettere di cui è composta (non importa se la parola ottenuta non è di senso compiuto).
Esempio*:** se da input sono state inserite le parole ***casa e ***sala***, si consideri la parola ***casa*** e si permutino in modo casuale le sue lettere. Dopo questo procedimento si potrebbe ad esempio ottenere la parola **asca**.

Esercizio. Si scriva in Python una funzione booleana **ricorsiva** che riceva una lista di numeri interi e restituisca True se le somme di due interi consecutivi nella lista sono alternatamente una pari ed una dispari, ossia comunque si prenda un intero x nella lista, se la somma di *x* con l'intero che lo precede è pari (risp. dispari), la somma di *x* con l'intero che lo segue deve essere dispari (risp. pari).
Esempio: la funzione invocata su una lista contenente la sequenza di interi 1 3 2 6 3 3 2 dovrà restituire True. Infatti le somme degli interi consecutivi sono 1+3 - pari, 3+2 – dispari, 2+6 – pari, 6+3 – dispari, 3+3 – pari, 3+2 – dispari.
Specificare quale deve essere la prima invocazione per la funzione.

Esercizio.
Ciccio Pasticcio è un grande appassionato di calcio e segue con grande interesse il torneo che annualmente viene organizzato nella sua città, e che vede coinvolte decine di squadre ognuna delle quali rappresenta un quartiere diverso. Il torneo è organizzato in maniera simile al campionato nazionale: tutte le squadre devono scontrarsi tra loro in due gironi, uno di andata ed uno di ritorno. Quindi, ogni coppia di squadre A e B si scontra due volte, una volta in cui A è la squadra di casa e B è la squadra ospite (ossia si gioca nel campo di calcio del quartiere della squadra A) e un'altra in cui B è la squadra di casa e A è l'ospite. Per essere sempre aggiornato sull'andamento del torneo, dopo ogni partita, Ciccio segna il risultato su un tabellone che ha appositamente predisposto, in cui le righe rappresentano le squadre che giocano in casa e le colonne le squadre ospiti; in corrispondenza di ogni coppia di squadre, Ciccio segna 1 se la squadra di casa ha vinto, 0 se ha pareggiato, 2 se ha perso.
Esempio La figura seguente mostra un esempio di tabellone per un torneo in cui sono coinvolte 4 squadre:

	S1	***S2***	***S3***	***S4***
S1	-	*2*	*0*	*1*
S2	*1*	-	*2*	*2*
S3	*1*	*1*	-	*2*
S4	*0*	*0*	*1*	-

Secondo il tabellone, quando la squadra ***s1*** ha giocato in casa, ha vinto contro ***s4***, pareggiato contro ***s3*** e perso contro ***s2***. Nelle partite in cui la squadra ***s4*** ha giocato fuori casa, ha perso contro ***s1*** e vinto contro ***s2*** ed ***s3***. Sulla diagonale principale sono presenti dei trattini ad indicare che non è previsto che una squadra giochi contro se stessa.

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410

Alla fine del torneo Ciccio "interroga" il tabellone per determinare la squadra vincitrice, ed altre informazioni relative all'andamento delle partite. Col passare degli anni però il torneo è diventato sempre più famoso, il numero di squadre che vi partecipano è cresciuto notevolmente e Ciccio, che come al solito, è un gran pasticcione, non riesce più a determinare correttamente le informazioni che gli servono. Conoscendo le tue abilità informatiche, Ciccio ti ha chiesto di sviluppare per lui un sistema che lo aiuti nel gestire il tabellone.

A tale scopo, si scriva un programma completo che, letto da input un "tabellone" determini:
- la squadra con il maggior numero di vittorie in casa* (nell'esempio precedente, è la squadra s3 che ha due vittorie in casa).
- la squadra che ha vinto il campionato, ossia la squadra che ha totalizzato più punti*, e quella che è arrivata ultima, cioè con meno punti*. Il punteggio di ogni squadra si calcola sommando i punti ottenuti ad ogni partita: 3 punti per ogni vittoria, 1 punto per ogni pareggio (nell'esempio precedente, la squadra arrivata prima è ***s4*** con 11 punti, mentre l'ultima è ***s1*** con 5 punti).

Variante 1: Si determini la squadra con il maggior numero di vittorie (sia in casa che fuori)*.
Variante 2: Si verifichi se esiste una squadra che ha perso tutte le partite del torneo.

* Nota: Se più di una squadra soddisfa la proprietà richiesta è sufficiente restituirne una (ad esempio la prima che si incontra durante la verifica)

Esercizio.
La tua amica Renata Limbranata è nei guai. Si è sposata solo pochi giorni fa eppure il suo matrimonio sembra già andare a rotoli. Suo marito passa la maggior parte del giorno al computer a scrivere mail di lavoro (dice lui!), ma lei è molto sospettosa, non riesce a credere che si sia portato il lavoro anche in viaggio di nozze… E così ha deciso! Approfittando delle assenze del marito ti ha inoltrato tutte le sue mail (circa un centinaio), pregandoti di leggerle e farle sapere se, come sospetta lei, il marito ha un'amante. Tu però, che conosci Renata e che sai quanto possa essere paranoica a volte, decidi di non sprecare belle giornate a leggere mail con molta probabilità noiose e di affidarti alle tue abilità informatiche per risparmiare tempo. A tale scopo, si scriva in Python un programma completo opportunamente modularizzato in funzioni che, dato in input l'elenco delle mail di Renata (che per semplicità si può implementare come lista di stringhe) conti quante sono le mail in cui è contenuta almeno una tra le seguenti parole: *amore, tesoro, cucciolotto, trottolino.*
Variante: Si conti il numero di mail che contengono contemporaneamente la parola *amore* e la parola *bacio* e non contengono la parola *moglie*.

Esercizio.
Sia **A** l'insieme **ricorsivo** di numeri naturali definito come segue:

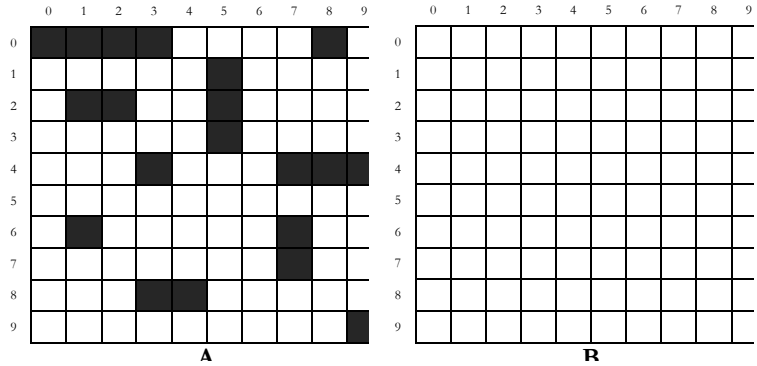
$$\left\{ \begin{array}{l} 0 \notin A \\ 1 \in A \\ x \in A \quad \Leftrightarrow \quad \frac{3x-2}{6} \in A, \quad \forall \quad x > 1 \end{array} \right\}$$

Si scriva in Python una funzione **ricorsiva** che verifichi se un dato numero appartiene o meno all'insieme **A**.

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410

Specificare quale deve essere la prima invocazione per la funzione.
NOTA: La divisione è intera.

Esercizio.



Si implementi in Python il famoso gioco **battaglia navale** tra due giocatori. Ogni giocatore ha a disposizione due griglie di uguale dimensione (stabilita all'inizio del gioco) che indicheremo con **A** e **B** ed una flotta di navi da disporre sulle griglie. Le navi sono di diversa grandezza e possono occupare da 1 a 4 caselle dello schema. La flotta di ogni giocatore è così composta: 4 sommergibili (navi da 1 casella), 3 vedette (navi da 2 caselle), due incrociatori (navi da 3 caselle), una portaerei (nave da 4 caselle). Prima di iniziare a giocare, ogni giocatore posiziona segretamente sulla propria griglia **A** la sua flotta. Le navi devono essere disposte orizzontalmente o verticalmente e non devono toccarsi tra loro (neppure in diagonale). La figura sopra mostra un esempio di disposizione corretta delle navi (come si può notare la griglia B è inizialmente vuota). Una volta terminata la disposizione delle navi, inizia il gioco vero e proprio: ogni giocatore a turno cercherà di colpire una nave dell'avversario con un colpo di cannone indicando le coordinate di una casella (ad esempio 6 , 7 indica la casella all'incrocio tra la riga 6 e la colonna 7). L'altro giocatore risponderà comunicando l'esito del colpo. In particolare, le risposte possibili sono "acqua" se la cannonata non ha colpito alcuna nave, "colpito" se la cannonata ha colpito una nave, ma ci sono altre caselle occupate dalla nave che ancora non sono state colpite, "affondato" se la cannonata ha colpito l'ultima casella ancora a "galla" di una nave.

A questo punto l'esito del colpo viene registrato in maniera automatica (ossia senza che siano i giocatori a farlo) sulle griglie. In particolare per il giocatore che ha subito il colpo, questa informazione viene memorizzata in maniera opportuna sulla griglia **A**, mentre per il giocatore che ha sparato, questa informazione viene invece memorizzata sulla griglia **B** (quella inizialmente vuota) che viene consultata dal giocatore al momento di decidere quale casella della griglia **A** avversaria colpire con una cannonata.

Il gioco termina quando uno dei due giocatori è riuscito ad affondare tutte le navi dell'avversario.
Per la disposizione delle navi di ogni giocatore si può utilizzare una funzione *distribuisciNavi* che si può supporre interamente implementata.
Variante: Si implementi la funzione *distribuisciNavi*, utilizzata nell'implementazione del gioco, per disporre in maniera corretta le navi sulla griglia.

Esercizio.
Scrivere un programma Python che legga da input un intero N e una frase F ed esegua la verifica descritta di seguito.
Sia L la lunghezza di F, si assuma che N sia strettamente minore di L.
Sia F1 la stringa formata dai caratteri alfanumerici che si trovano in F a partire dalla posizione 0 alla posizione N-1.

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 - fax (+39) 0984 496410

Sia F2 la stringa formata dai caratteri alfanumerici che si trovano in F a partire dalla posizione N alla posizione L-1.
Verificare se F1 ed F2 sono una l'anagramma dell'altra, ovvero se tutti i caratteri che compongono F1 sono in F2 e viceversa. Stampare "SI" se tale verifica ha esito positivo, "NO" altrimenti.
Esempio. Se F fosse "casa cosa occa" e N fosse 7, F1="casaco", F2="saocca" il programma dovrebbe stampare "SI". Se N fosse 2, F1="ca" e F2="sacosaocca", il programma stamperebbe "NO".
Variante: Realizzare tale verifica attraverso una funzione **ricorsiva**.

Esercizio.
Si implementi una funzione pyhton che dati una frase A in cui le parole sono separate l'una dall'altra da uno spazio bianco e un lista di interi positivi B restituisca True se i numeri contenuti in B corrispondono esattamente alle lunghezze di tutte le parole contenute in A, rispettando anche l'ordine in cui queste appaiono in A, e restituisca False altrimenti.
Esempi:
Se A è la frase "oggi non si studia" e B è la lista 4 3 2 6 la funzione restituisce True.
Se A è la frase "oggi non si studia" e B è la lista 4 6 2 3 la funzione restituisce False.
Se A è la frase "oggi non si studia" e B è la lista 4 3 2 6 8 la funzione restituisce False
Variante: Realizzare tale verifica attraverso una funzione **ricorsiva**.

Esercizio. Nel paese di Magicolandia si stanno per svolgere gli Esami di Regno per Apprendista Stregone Aggiunto di Categoria Xilla. Una delle prove consiste nello scrivere una formula magica composta da parole piuttosto strane (salacadula, bidibibodibibbu', simsalallà, abracardabrata, bumshakalaka, etc.). Il nostro amico Harry Pastitcher, come Gran Stregone Maximo, è il presidente di commissione, e dovrà valutare gli elaborati. Negli ultimi anni gli studenti, sempre più pigri, hanno iniziato un'attività esagerata di collaborazionismo sottobanco. Per velocizzare la scoperta di candidati che hanno "copiato", ci ha chiesto di realizzare un semplice programma scritto in Python che confronti due testi e conti quante sono le parole magiche in comune.
Nel dettaglio, si scriva un programma in Python che, lette da input due frasi e in cui le parole sono separate da uno spazio, restituisca il numero di parole che esse hanno in comune.

Esercizio.
Si scriva in Python una funzione RICORSIVA che, ricevuti come parametri (almeno) tre liste di caratteri, A B e C, e la loro dimensione, riempia C *incastrando ad incrocio* i caratteri presenti in A e B come descritto di seguito. Ogni coppia di elementi consecutivi in C deve essere formata da un carattere di A e da uno di B; gli elementi di A vanno usati da sinistra verso destra(cioè prima il primo, poi il secondo e così via) mentre gli elementi di B devono essere usati da destra verso sinistra (ossia, prima l'ultimo, poi penultimo, ecc.). Pertanto, la prima coppia di elementi in C sarà formata dal primo carattere di A e dall'ultimo in B, la seconda coppia in C, conterrà, il secondo elemento di A ed il penultimo di B e così via, finché tutti i caratteri di A e di B saranno stati considerati.
Si noti che, perché l'incastro sia fattibile, le liste A e B devono avere la stessa dimensione *n*, mentre la dimensione di C deve essere 2**n*. Se la funzione riceve come parametri liste le cui dimensioni non rispettano queste condizioni, l'incastro non deve essere effettuato e la funzione deve terminare restituendo false.

A	V	E	R	D	E

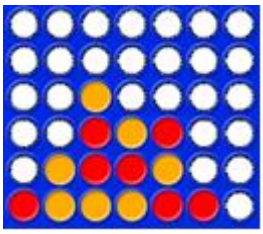
B	*R*	*O*	*S*	*S*	*O*

C	V	*O*	E	*S*	R	*S*	D	*O*	E	*R*

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 - fax (+39) 0984 496410

ESEMPIO: Date le liste A e B riportate sopra, la funzione dovrà riempire la lista C come illustrato in figura e terminerà restituendo true.

Esercizio.



Si implementi in Python il famoso gioco "FORZA 4" per due giocatori. Si riporta di seguito una breve descrizione del gioco. Si gioca su una griglia di dimensioni NxM. L'obiettivo di ciascun giocatore è mettere in fila quattro proprie pedine, in orizzontale, verticale o diagonale; l'elemento fondamentale del gioco è la forza di gravità: la scacchiera è infatti posta in verticale e le pedine vengono fatte cadere lungo una griglia verticale, in modo tale che una pedina inserita in una certa *colonna* vada sempre a occupare la posizione libera situata più in basso nella colonna stessa. I due giocatori si alternano, e ciascuno può inserire una pedina sola durante il proprio turno. Il gioco va avanti fino a quando uno dei due giocatori non vince (riuscendo quindi ad allineare quattro pedine) oppure non c'è più spazio per inserire nuove pedine (la griglia è tutta piena senza che nessuno abbia allineato almeno 4 pedine): in questo caso si ha un pareggio.
Il programma deve gestire il gioco portato avanti da 2 umani. Deve chiedere A TURNO in quale colonna il giocatore corrente desidera inserire la propria pedina, e deve automaticamente determinare in quale riga si deve finire (come farebbe nella realtà, seguendo la forza di gravità). Ogni volta che un giocatore inserisce una pedina, il programma dovrà anche determinare se per caso egli è il vincitore (cioè se la pedina che ha inserito ha determinato una fila di 4) oppure se la griglia è stata riempita e c'è il pareggio. In tal caso il programma termina, ovviamente dopo aver comunicato ai giocatori l'esito del gioco.
NOTA: per semplicità, si consideri una versione semplificata del gioco, in cui si vince solo se le pedine sono in fila ORIZZONTALE o VERTICALE (in pratica si può omettere il controllo sulle diagonali).
Variante: Si implementi il controllo che verifica se un giocatore ha vinto anche controllando eventuali pedine dello stesso colore in fila lungo le diagonali.

Esercizio.
Una tabella di dimensione 12*31 tiene traccia di tutte delle spese sostenute da Ciccio Pasticcio nell'arco del 2018. Scrivere un programma Python che utilizzando opportune funzioni, implementi tutte le seguenti funzionalità:

- calcolare il mese in cui Ciccio ha speso di più;
- verificare se esiste un giorno x tale che in tutti i mesi, la spesa sostenuta nel giorno x è sempre uguale;
- verificare se esiste un mese M per il quale esistono 3 giorni consecutivi tali che i primi due hanno spesa complessiva superiore alla media delle spese di M e per il terzo giorno la spesa è 0.

Utilizzare la ricorsione per lo svolgimento di uno dei punti su indicati (a scelta).

Esercizio
Si implementi una funzione Python che ricevuti in input una lista A di caratteri di dimensione N, una lista B di interi di dimensione N-1, una matrice (lista di liste) C di caratteri di dimensione L x M, scorrendo la matrice da destra a sinistra e dall'alto al basso, individui se ciascun carattere A[i] di A occorre in C ed il

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410

successivo carattere si trova a distanza B[i]. In tal caso, si stampino i caratteri della matrice che stanno tra i caratteri di A, altrimenti si stampi "ERRORE".
N.B Lo svolgimento corretto dell'esercizio senza l'utilizzo della ricorsione vale metà punteggio.
Esempio 1. Se abbiamo una lista A = {c,a,s,a,l,e}, una lista B = {6,2,3,2,1} e una matrice C come segue:

m	n	c	**i**	**l**	**m**
i	**o**	a	**p**	s	**a**
p	a	**a**	l	e	c
v	b	w	s	x	u

Il programma stamperà: ilmiopapa
Infatti

- il primo carattere di A, **c**, si trova in posizione [0][2];
- a distanza 6, troviamo il secondo carattere di A, **a**;
- a distanza 2, da quest'ultimo, troviamo il terzo carattere di A, **s**;
- a distanza 3, da quest'ultimo, troviamo il quarto carattere di A, **a**;
- a distanza 2, da quest'ultimo, troviamo il quinto carattere di A, **l**;
- a distanza 1, da quest'ultimo, troviamo il sesto carattere di A, **e**.

Quindi la verifica è sodisfatta e i caratteri compresi tra i caratteri di A formano **ilmiopapa.**

Esercizio.
Si scriva in Python un programma completo opportunamente modularizzato in funzioni che simuli una variante del gioco dell'impiccato tra due giocatori. Le regole del gioco sono le seguenti: il primo giocatore sceglie una parola segreta e mostra all'altro una serie di trattini (lunga quanto la lunghezza della parola da indovinare), ognuno dei quali nasconde una lettera. Il secondo giocatore ha a disposizione sette tentativi, in cui può dire qual è secondo lui la parola segreta oppure dichiarare una lettera; se la lettera dichiarata è presente nella parola segreta, il primo giocatore scopre tutti i trattini che nascondono la lettera indovinata. Il gioco termina quando il secondo giocatore indovina la parola segreta (e quindi vince) oppure esaurisce i tentativi a disposizione (e quindi vince il primo giocatore).
Esempio di partita: Il giocatore 1 sceglie come parola segreta **portobello** e mostra quindi la serie di trattini: _ _ _ _ _ _ _ _ _ _.
1° tentativo il giocatore 2 sceglie di dichiarare la lettera **o** ed il giocatore 1 mostra: **_o_ _o_ _ _ _o.**
2° tentativo il giocatore 2 sceglie di dichiarare la lettera **s** ed il giocatore 1 mostra: **_o_ _o_ _ _ _o.**
3° tentativo il giocatore 2 sceglie di dichiarare la lettera **l** ed il giocatore 1 mostra **_o_ _o_ _ l l o**.
4° tentativo il giocatore 2 sceglie di provare ad indovinare direttamente la parola e dice **portogallo**. Il giocatore 1 comunica che la risposta è sbagliata.
5° tentativo il giocatore 2 sceglie di nuovo di provare ad indovinare la parola e dice **portobello**. Il giocatore 1 comunica che la risposta è esatta ed il gioco termina.

Esercizio.
Si scriva in Python una funzione **ricorsiva** che riceva una lista di numeri interi e la sua dimensione (che si suppone essere dispari) e restituisca True se la lista è costituita da una serie di picchi ravvicinati. Un picco è un gruppo di tre elementi consecutivi in cui quello centrale è maggiore dei due esterni e due picchi si dicono ravvicinati se il terzo elemento di un picco è il primo elemento del picco successivo.

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 - fax (+39) 0984 496410

Esempio: la funzione invocata su una lista contenente la sequenza di interi 1 3 2 4 -3 7 -5 8 4 dovrà restituire True. Infatti la sequenza contiene i 4 picchi {1, 3 , 2}, {2, 4, -3}, {-3, 7, -5}, {-5, 8, 4} e sono ravvicinati (ad esempio, i picchi {1, 3 , **2**}e {**2**, 4, -3}condividono il numero **2**, e così via).
Specificare quale deve essere la prima invocazione per la funzione.

Esercizio. Si scriva un programma completo che simuli il gioco con le carte napoletane "Asino" tra due giocatori. Le regole del gioco sono le seguenti:

1) Dal mazzo di carte (composto da 40 carte, 4 semi, 10 carte per ogni seme) viene scartato uno dei cavalli (la carta numero nove) e le rimanenti 39 carte vengono distribuite tra i due giocatori, un giocatore avrà quindi 20 carte in mano, mentre l'altro ne avrà 19.
2) Alla prima mano, i due giocatori scartano tutte le coppie di carte con lo stesso valore, ad esempio (due otto, due cinque, etc.), per cui dopo questa fase ogni giocatore si troverà con un numero di carte sicuramente inferiore di quello iniziale.
3) A questo punto inizia il gioco vero e proprio. A turno ogni giocatore prende **a caso** una carta tra quelle dell'altro e controlla se, dopo la presa, si è formata una nuova coppia (ad esempio, il giocatore ha in mano un due di bastoni e prende dall'altro giocatore un due di coppe). Se questo avviene, il giocatore scarta la coppia e si ritroverà con due carte in meno. Se invece, dopo la presa, non si forma una nuova coppia, il giocatore si ritroverà con un carta in più.
4) Il gioco termina quando uno dei due giocatori rimane senza carte in mano vincendo la partita. L'altro giocatore rimarrà invece con una sola carta in mano (un cavallo) e sarà perciò l'asino del gioco.

Suggerimenti:
(i) Per la rappresentazione delle carte di ogni giocatore, si potrebbe utilizzare una matrice di valori booleani, in cui le righe sono i 4 semi mentre le colonne rappresentano i 10 possibili valori delle carte. Se nella matrice, nella posizione con un indice di riga i e colonna j c'è True allora la carta numero j con seme i è presente tra le carte del giocatore, altrimenti non lo è
(ad esempio, supponendo di indicare con C1 la matrice corrispondente alle carte del primo giocatore , se C1[2][3] == True allora la carta 3 con seme 2 è presente tra le carte del primo giocatore).
(ii) Per controllare l'andamento del gioco si potrebbero tenere dei contatori per il numero di carte di ogni giocatore.
(iii) La distribuzione delle carte tra i due giocatori deve avvenire in modo random.

Esercizio.
Si consideri il seguente rompicapo enigmistico: all'interno di una frase F potrebbe essere celata una parola segreta; per individuarla è necessario applicare le seguenti regole: 1) cancellare da F tutte e sole le occorrenze delle lettere che compaiono più di una volta in F, 2) se nessuna lettera è rimasta dopo la cancellazione, allora nella frase non è nascosta alcuna parola segreta; altrimenti 3) la parola nascosta si ottiene considerando tutte le lettere rimaste (tralasciando ovviamente gli spazi), prese nell'ordine.

Si scriva un programma OPPORTUNAMENTE MODULARIZZATO IN FUNZIONI che letta da input un frase F verifichi se in F è nascosta o meno una parola, secondo il meccanismo descritto sopra; in caso affermativo la stampi su standard output, altrimenti stampi il messaggio "Nessuna Parola Nascosta". NOTA: per semplicità, si può supporre che la frase contenga solo lettere maiuscole e spazi. Inoltre, non è necessario verificare che la parola segreta sia di senso compiuto. ESEMPIO: Se la frase introdotta da

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410

input fosse IO NON STIMO ME STESSO, il programma dovrebbe stampare su standard output "Nessuna Parola Nascosta"; se invece la frase in input fosse UNGERE CHIARE CERE, il programma dovrebbe stampare "La parola segreta e' UNGHIA".

Esercizio. Si scriva una funzione RICORSIVA che, ricevuti come parametri (almeno) una lista di numeri interi e la sua dimensione, verifichi se sono presenti almeno tre elementi in posizioni distinte di cui uno è pari al prodotto degli altri due, ed in caso positivo, restituisca le loro posizioni. Se sono presenti più triple che soddisfano la proprietà, è necessario restituire le posizioni della prima. ESEMPIO: dato la lista qui di fianco, di dimensione 10, la funzione dovrà restituire le posizioni 1, 2 e 4, infatti gli elementi ad esse corrispondenti (8, 4 e 2 rispettivamente), sono tali per cui 4*2=8.

1	***8***	**4**	*3*	**2**	*6*	5	*9*	10	*11*

Esercizio.
Data una matrice **M** di dimensione **m*n**, una *banda orizzontale* di dimensione **x,** con **1=<x<=m**, è un insieme di **x** righe consecutive tutte uguali fra loro; si noti che una riga diversa sia dalla precedente che dalla successiva, fa parte di una banda di dimensione 1. Una banda orizzontale che include la prima riga della matrice viene detta *banda pivot*. Una matrice si dice a bande orizzontali, se tutte le bande individuabili sono di dimensione pari a quella della banda pivot. Si scriva una funzione che ricevuto come parametro (almeno) una matrice **M** di dimensione **m*n**, verifichi se la matrice è a bande orizzontali, e restituisca la dimensione delle bande. **NOTA:** Per la verifica è necessario dapprima calcolare la dimensione della banda pivot. Inoltre, una matrice in cui tutte le righe sono diverse tra loro, è una matrice a bande orizzontali di dimensione 1.
ESEMPIO: la matrice 7*6 riportata in basso a sinistra, non è una matrice a bande orizzontali: infatti, è possibile individuare, oltre alla banda pivot di dimensione 2, altre 2 bande orizzontali di dimensione 1 e 3 rispettivamente (entrambe di dimensione diversa da quella della banda pivot). Invecw, la matrice 7*6 riportata in basso a destra è a bande orizzontali di dimensione 2.

1	6	3	4	8	3	9
1	6	3	4	8	3	9
2	7	3	5	5	4	2
5	2	7	8	4	5	1
5	2	7	8	4	5	1
5	2	7	8	4	5	1

Banda di dimensione 2 (PIVOT)
Banda di dimensione 1
Banda di dimensione 3

1	6	3	4	8	3	9
1	6	3	4	8	3	9
2	7	3	5	5	4	2
2	7	3	5	5	4	2
5	2	7	8	4	5	1
5	2	7	8	4	5	1

Banda di dimensione 2 (PIVOT)
Banda di dimensione 2
Banda di dimensione 2

<u>Esercizio.</u>
Si consideri la successione dei numeri positivi dispari, 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31… e si applichi il seguente processo di eliminazione:

- Il primo numero più grande di 1 è il 3. Si considerino quindi gli elementi a gruppi di 3 consecutivi e in ogni gruppo si elimini il terzo numero; la nuova successione sarà 1,3,7,9,13,15,19,21,25,27,31…
- Il primo numero più grande di 3 è il 7. Si considerino quindi gli elementi a gruppi di 7 consecutivi e in ogni gruppo si elimini il settimo numero; la nuova successione sarà 1,3,7,9,13,15,21,25,27,31….
- Il primo numero più grande di 7 è il 9. Si considerino quindi gli elementi a gruppi di 9 consecutivi e in ogni gruppo si elimini il nono numero; la nuova successione sarà 1,3,7,9,13,15,21,25,31…. E così via.

I numeri che rimangono dopo l'applicazione del processo, vengono detti numeri **fortunati**.

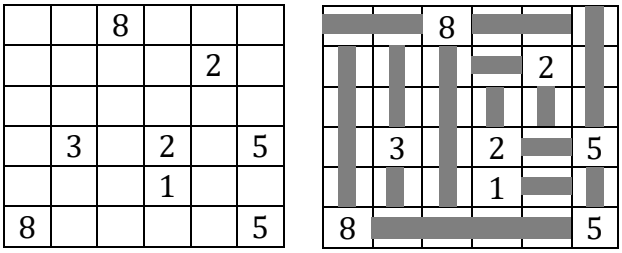
CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410

Si scriva un programma, che letta da input una lista contenente i numeri naturali dispari da 1 ad N, utilizzando la RICORSIONE, modifichi la lista affinché questo contenga solo i numeri fortunati minori di N. Ad esempio, se N è pari a 30, la lista dovrà contenere solo i numeri 1,3,7,9,13,15,21,25,27.

Esercizio.
Si consideri il seguente rompicapo, denominato WALLS: viene data una griglia quadrata di dimensione N*N; alcune celle della griglia sono vuote, altre invece contengono dei numeri. Lo scopo del gioco è riempire tutte le celle vuote della griglia con dei segmenti orizzontali o verticali, così da creare dei muri, secondo le seguenti indicazioni:

- I muri nascono dalle celle che contengono i numeri.
- Se una cella C contiene un numero X, allora la lunghezza totale dei muri che nascono da C deve essere X; ad esempio, da una cella contenente il numero 7 potrebbe partire un muro verticale di lunghezza 3 (ossia che occupa tre caselle) verso nord, un muro orizzontale di lunghezza 2 verso destra, e un muro verticale di lunghezza 2 verso sud.
- Un muro termina quando incontra il bordo, un altro numero, o un muro orientato diversamente.

Le griglie seguenti mostrano uno schema di WALLS (sinistra) e la rispettiva soluzione (destra).



Si noti come tutte le celle originariamente vuote, ora contengano un pezzo di muro, orizzontale o verticale, e come i muri rispettino le indicazioni fornite dai numeri: ad esempio, dal numero 8 nella prima riga parte un muro che occupa due celle verso sinistra, un muro che occupa due celle verso destra, e un muro che occupa 4 celle verso il basso, per una lunghezza totale pari appunto ad 8. Si noti inoltre come un muro possa essere "condiviso" da due celle numerate; ad esempio guardando l'ultima riga in basso, si vede come il muro orizzontale di lunghezza 4 vada contato sia come muro che parte dall'8 in basso a sinistra, sia come muro che parte dal 5 in basso a destra.

Implementare un programma che, verifichi che una possibile soluzione per uno schema di Walls, fornita da input, sia corretta. In pratica il programma deve chiedere all'utente di inserire da input una griglia già riempita, e il programma verifica che il riempimento rispetti le condizioni descritte precedentemente. **SUGGERIMENTO:** Si consiglia di rappresentare i pezzi di muro orizzontale con il numero -1 e i pezzi di muro verticale con il numero -2. Nel caso dell'esempio precedente, la griglia in input sarebbe quella illustrata a destra.

-1	-1	**8**	-1	-1	-2
-2	-2	-2	-1	**2**	-2
-2	-2	-2	-2	-2	-2
-2	**3**	-2	**2**	-1	**5**
-2	-2	-2	**1**	-1	-2
8	-1	-1	-1	-1	**5**

CAMPUS DI ARCAVACATA http://www.unical.it/portale/strutture/dipartimenti_240/matinf/
87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 - fax (+39) 0984 496410