Il metodo di cifratura di Vigenère è stato, fino alla fine del XIX secolo, uno dei principali metodi per la conservazione o la trasmissione di testi (o messaggi) in un formato segreto (*cifrato*) ed incomprensibile ad altri tranne che al legittimo proprietario o destinatario.
Il metodo utilizza a tal fine due elementi fondamentali:

- una tabella di 26 righe, in cui ogni riga è costituita da un alfabeto (inglese, di 26 lettere), via via scalato di una lettera a sinistra rispetto alla riga precedente (si veda la figura successiva);
- una parola chiave, scelta da chi opera la trasformazione del testo (*cifratura*) e da non divulgare ad alcuno, tranne all'eventuale destinatario del testo stesso.

Il metodo viene qui illustrato con un esempio. Si supponga di voler trasformare in un formato segreto (*cifrare*) il seguente testo:

```
Per essere promossi agli esami bisogna prendere almeno 18. Gli
studenti bravi ottengono 30. Quelli ancora migliori
prendono 30 e lode.
```

Si supponga di avere scelto la seguente parola chiave: TRIESTE

La tabella con gli alfabeti è la seguente (per l'interpretazione dei commenti si veda il seguito):

Per eseguire la trasformazione del testo, si immagini di aver scritto sotto ogni riga del testo stesso la parola chiave, ripetuta di seguito, in modo da associare ad ogni lettera del testo la lettera della parola chiave che compare sotto di essa:

```
Per essere promossi agli esami bisogna prendere almeno 18. Gli
TRIESTETRIESTETRIESTETRIESTETRIESTETRIESTETRIESTETRIESTETRIEST
studenti bravi ottengono 30. Quelli ancora migliori
ETRIESTETRIESTETRIESTETRIESTETRIESTETRIESTETRIESTETRIESTET
prendono 30 e lode.
RIESTETRIESTETRIEST
```

Per ogni singola lettera del testo, si cerchi ora nella prima colonna della tabella tale lettera e nella prima riga la lettera della parola chiave ad essa associata (cioè la lettera della parola chiave scritta sotto la lettera del testo). Restano così individuate rispettivamente una riga ed una colonna della tabella. La lettera che si trova all'incrocio tra tale riga e tale colonna va a far parte del testo cifrato. Ad esempio, il testo sopra riportato inizia con la lettera "P", sotto cui appare la lettera "T" della parola chiave. Nella tabella, in corrispondenza della riga che inizia per "P" e della colonna che inizia per "T", compare la lettera "I", che viene a costituire la prima lettera del testo cifrato (si vedano i commenti accanto alla tabella nel foglio precedente). La seconda lettera del testo è una "e", a cui è associata una "R". Nella tabella, in corrispondenza della riga iniziante per "E" e della colonna iniziante per "R", si trova una "V", che viene a costituire la seconda lettera del testo cifrato. Procedendo in tal modo per ogni lettera del testo originario, si ottiene infine il testo cifrato:

```
IVZ WLWXIM HKSFFAWA EZCQ WLEFZ FALSZEI HKIGUMVW EEDMRG 18. KDB
WMLLIFMM SZENB HKBIFZSGF 30. JLMPDB TEKSJT FZOPAHVB
GZIFWSGF 30 I CWHW.
```

NB: la trasformazione avviene solamente sulle lettere, mentre spazi, numeri, punteggiatura, ecc. vengono mantenuti invariati. Inoltre, per semplicità, si supponga che il testo originario non contenga caratteri accentati e non si faccia distinzione tra lettere minuscole e maiuscole nel testo trasformato e nella parola chiave.

Per riottenere di nuovo il testo originale a partire dal testo cifrato (supponendo nota la parola chiave), si immagini di scrivere sotto al testo cifrato la parola chiave ripetuta e si operi in modo inverso: si cerchi la lettera della chiave nella prima riga, si scorra la colonna così individuata fino a trovare la lettera del testo cifrato e ci si sposti a sinistra nella tabella, lungo la riga, fino ad individuare la lettera del testo originale.

Si chiede di realizzare un progetto Visual Basic che effettui quanto segue:

- consenta all'utente di indicare il nome di un file testo contenente il testo da trasformare;
- consenta all'utente di indicare se desidera trasformare il testo contenuto nel file cifrandolo secondo il metodo di Vigenère o rendere nuovamente leggibile un testo già cifrato con tale metodo e contenuto nel file stesso;
- chieda all'utente di indicare la parola chiave;
- effettui la trasformazione richiesta scrivendo un nuovo file testo.

Si supponga di aver a disposizione un file di testo di tre righe, in cui ogni riga contiene, nell'ordine, tre numeri reali, che rappresentano i coefficienti a, b, c di una retta nel piano di equazione $ax+by+c=0$.

Si richiede di scrivere un progetto Visual Basic che effettui quanto segue:

- consenta all'utente di indicare il file di testo
- verifichi che nessuna delle tre rette così individuate sia parallela o coincidente con una delle rimanenti;
- in caso affermativo, determini il punto di intersezione di ogni coppia di rette;
- nel caso in cui i punti così individuati siano distinti, calcoli il valore del perimetro del triangolo di vertici tali punti.

Suggerimento: si ricorda che, date due rette di equazione $a_1x+a_2y+a_3=0$ e $b_1x+b_2y+b_3=0$, queste sono parallele o coincidenti se e soltanto se esiste k reale tale che $a_1=kb_1$ e $a_2=kb_2$, oppure, in forma equivalente, se e soltanto se $a_1b_2=a_2b_1$.

Si supponga di avere a disposizione il file di testo contenente informazioni relative alle quantità disponibili di alcuni generi alimentari. Ogni riga di tale file di testo contiene le informazioni relative ad uno di tali generi alimentari, secondo il seguente schema:

1) codice del genere alimentare: numerico intero (campo chiave)
2) denominazione del genere alimentare: stringa
3) quantità disponibile: numerica

Tale file è ordinato secondo il codice in senso crescente.

Un altro file di testo contiene invece gli ingredienti di una ricetta.
Il file contiene un certo numero di righe, una per ogni ingrediente necessario, la cui struttura è di seguito indicata:

1) codice del genere alimentare: numerico intero (campo chiave)
2) quantità del genere alimentare necessaria per la ricetta: numerica

Si assuma che per le quantità indicate esista una convenzione relativa alle unità di misura (liquidi misurati in litri, ...).

Si chiede di realizzare un programma Visual Basic che consenta all'utente di indicare i due file e che, letti dal secondo di questi gli ingredienti della ricetta, visualizzi la situazione aggiornata delle quantità disponibili qualora si sia realizzata la ricetta stessa, sottraendo cioè dalle quantità disponibili le quantità che verrebbero impiegate nella realizzazione.
Qualora per un qualche ingrediente della ricetta non vi sia sufficiente quantità disponibile, o lo stesso ingrediente non sia affatto disponibile, il programma dovrà visualizzare una opportuna segnalazione.

Il programma dovrà poi salvare la situazione aggiornata delle quantità disponibili in un file.

Si supponga di avere a disposizione un file testo contenente su ogni riga, nell'ordine, le seguenti informazioni relative a sentieri di montagna:

- località di inizio del sentiero: stringa;
- località di arrivo del sentiero: stringa;
- numero del sentiero: numerico intero.

Il file è ordinato in senso crescente in base a località di inizio, località di arrivo e numero del sentiero, nell'ordine.
È inoltre a disposizione un altro file di testo contenente su ogni riga, nell'ordine, le seguenti informazioni:

- numero del sentiero: numerico intero;
- dislivello in salita in metri presente lungo il sentiero: numerico intero;
- dislivello in discesa in metri presente lungo il sentiero: numerico intero.

Il file è ordinato in senso crescente in base al numero di sentiero.
Si scriva un progetto Visual Basic che consenta all'utente di indicare una località di inizio e visualizzi tutte le località di arrivo raggiungibili direttamente da tale località di inizio percorrendo un sentiero contenuto nel file dei sentieri. Inoltre, per ognuna di queste località di arrivo dovrà essere visualizzato, in ore e minuti, il tempo (eventualmente approssimato) necessario a percorrere il sentiero, assumendo convenzionalmente che in un'ora si percorra un dislivello di 300 metri in salita o un dislivello di 600 metri in discesa.

Si abbiano a disposizione tre file, di seguito convenzionalmente denominati STUDENTI, MATERIE e PUNTI .

Ogni record del file STUDENTI è costituito da campi contenenti le seguenti informazioni:

1) matricola dello studente: numerico intero
2) nome dello studente: stringa
3) cognome dello studente: stringa
4) numero *n* di esami sostenuti (al massimo 10): numerico intero
5) un array di dieci elementi contenenti informazioni relative agli esami sostenuti dallo studente: array di

I primi *n* elementi di tale array contengono ciascuno le seguenti informazioni:

a) codice della materia: numerico intero
b) voto conseguito nell'esame relativo alla materia (da 1 a 30): numerico intero
c) il carattere N in assenza di lode, il carattere S in presenza di lode (quest'ultimo può essere presente solo eventualmente se il voto conseguito è 30)

Ogni record del file MATERIE è costituito dalle seguenti informazioni:

1) codice della materia: numerico intero
2) denominazione della materia: stringa
3) il carattere O se l'esame relativo alla materia si svolge solamente con una prova orale, il carattere S se tale esame comporta una prova scritta

Il file MATERIE contiene soltanto 10 record ed i suoi record sono ordinati in ordine crescente di codice.

Il file PUNTI contiene 31 elementi, ognuno dei quali è un intero compreso tra 0 e 5. Tali interi rappresentano punteggi che vengono assegnati per concorrere ad una borsa di studio e sono attribuiti agli studenti in base alle votazioni conseguite nei singoli esami, in modo che una votazione pari a *i* conseguita in un esame comporta per lo studente l'acquisizione di un numero di punti pari al valore contenuto nell'*i*-esimo record di tale file. Il 31° record contiene il punteggio assegnato al risultato di 30 e lode.

Si chiede di costruire un programma Visual Basic che produca un file testo contenente, per ogni studente presente nel file STUDENTI, le seguenti informazioni:

1) matricola
2) cognome
3) nome
4) media dei voti conseguiti negli esami con prova solo orale
5) media dei voti conseguiti negli esami con prova scritta
6) media complessiva dei voti conseguiti negli esami

Si richiede inoltre che il programma produca un file testo contenente, per ogni studente presente nel file STUDENTI, le seguenti informazioni:

1) totale complessivo dei punti per la borsa di studio acquisiti attraverso gli esami sostenuti
2) matricola
3) cognome
4) nome

Tale file deve essere ordinato in ordine descrescente di punteggio acquisito.

Si supponga di avere a disposizione un file di testo, qui convenzionalmente chiamato **treni**, contenente i dati relativi ad alcuni viaggi in treno tra città italiane. Le informazioni contenute in ogni riga del file sono, nell'ordine, le seguenti:

- Numero del treno: numerico intero
- Località di partenza: stringa
- Località di arrivo: stringa
- Orario di partenza: stringa di 4 caratteri numerici nel formato HHMM, in cui cioè i primi due caratteri rappresentano l'ora ed i secondi due i minuti
- Orario di arrivo: come per l'orario di partenza
- Data di partenza: stringa di 8 caratteri numerici nella forma AAAAMMGG, in cui cioè i primi quattro caratteri rappresentano l'anno, i secondi due il mese e gli ultimi due il giorno
- Costo del viaggio in euro: currency

Il file è ordinato secondo il numero del treno e, a parità di numero di treno, per data.
Si richiede di scrivere un progetto Visual Basic che consenta all'utente di indicare il numero di un treno ed una data. Nel caso che il numero e la data inseriti corrispondano ad un viaggio presente nel file, il programma dovrà consentire successivamente all'utente di indicare il ritardo del treno all'arrivo in minuti. Il programma dovrà quindi visualizzare il numero di tale treno, le rispettive località di partenza e di arrivo, la data del viaggio, il ritardo ed il rimborso che verrà concesso al viaggiatore. Tale rimborso è pari a un ventesimo del costo del viaggio per ogni minuto di ritardo, con un massimo pari al costo del viaggio stesso.

Si supponga che un'azienda abbia a disposizione tre file, di seguito denominati per convenzione CLIENTI, ORDINI e PREZZI.

Il file CLIENTI contiene informazioni relative ai clienti dell'azienda. Ogni record di tale file contiene le seguenti informazioni:

1) codice del cliente: numerico intero
2) denominazione del cliente: stringa
3) importo relativo alla situazione creditoria (se positivo) o debitoria (se negativo) dell'azienda verso il cliente: numerico currency

Tale file è ordinato per codice cliente in ordine crescente.

Il file ORDINI contiene informazioni riguardanti gli ordini relativi a prodotti dell'azienda effettuati da alcuni clienti. Ogni record di tale file contiene le seguenti informazioni:

1) codice del cliente che ha effettuato l'ordine: numerico intero
2) codice del prodotto ordinato: numerico intero
3) numero di unità di prodotto ordinate: numerico intero

È ovviamente possibile che il file contenga più ordini da parte di un medesimo cliente. Tale file è ordinato per codice cliente in ordine crescente e, a parità di codice cliente, per codice prodotto in ordine anch'esso crescente.
Qualora il numero di unità indicate in un ordine sia negativo, l'ordine va inteso come una resa del prodotto da parte del cliente.

Il file PREZZI contiene i prezzi dei prodotti dell'azienda ed ogni suo record contiene le seguenti informazioni:

1) codice del prodotto: numerico intero
2) prezzo per unità di prodotto: currency

Si supponga ora che gli ordini contenuti nel file ORDINI vengano evasi. Si chiede quindi di costruire un programma Visual Basic che aggiorni, nel file CLIENTI, gli importi rappresentanti la situazione creditoria o debitoria dell'azienda verso i clienti in base al numero di pezzi ordinati o resi da ogni singolo cliente, indicati nel file ORDINI, facendo riferimento ai prezzi indicati nel file PREZZI, sia in caso di acquisto sia in caso di resa.

Esercizio P7.10. *Codici a barre postali.* Per ripartire più velocemente le lettere, il Servizio Postale degli Stati Uniti, l'United States Postal Service, incoraggia le aziende che spediscono grossi volumi di posta ad usare un codice a barre per indicare lo ZIP, il codice di avviamento postale (osservate la Figura 9).

Lo schema di codifica per un codice ZIP di cinque cifre è illustrato nella Figura 10. A ciascuna estremità, si trova una barra di delimitazione ad altezza intera. Le cinque cifre codificate sono seguite da una cifra di controllo, calcolata nel modo seguente: si sommano tutte le cifre e si sceglie la cifra di controllo che, sommata al totale, restituisce un multiplo di dieci. Per esempio, la somma di tutte le cifre del codice ZIP 95014 è uguale a 19, quindi la cifra di controllo è uno, perché porta il totale a 20.

Ciascuna cifra del codice ZIP, e la stessa cifra di controllo, sono codificate secondo lo schema della tabella seguente, in cui lo zero indica una mezza barra e l'uno una barra ad altezza intera:

	7	4	2	1	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

Notate che le codifiche per ciascuna cifra rappresentano tutte le combinazioni possibili di due barre intere con tre mezze barre. La cifra nel codice a barre si può calcolare facilmente utilizzando i pesi attribuiti alle colonne 7, 4, 2, 1 e 0. Per esempio, la codifica 01100 equivale a $0 \times 7 + 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times 0 = 6$. L'unica eccezione è zero, che, decodificato mediante la formula dei pesi, restituirebbe 11.

Figura 9
Un codice a barre postale

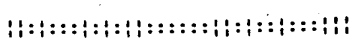
```
********** ECRLOT ** CO57

CODE C671RTS2
JOHN DOE                                CO57
1009 FRANKLIN BLVD
SUNNYVALE      CA 95014 - 5143
```

Figura 10
La codifica di un codice a barre di cinque cifre

—

Scrivete un programma che chieda all'utente di inserire un codice ZIP e che stampi il relativo codice a barre. Usate i due punti : per indicare le mezze barre, e il simbolo ¦ per le barre intere. Per esempio, il codice 95014 diventa:



Esercizio P11.15. *Quadrati magici.* Una matrice $n \times n$ riempita con i numeri 1, 2, 3, ..., n^2 è un quadrato magico, se la somma degli elementi di ogni riga, di ogni colonna e delle due diagonali ha lo stesso valore. Per esempio,

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Scrivete un programma che legga n^2 valori dalla tastiera e verifichi se formano un quadrato magico quando sono disposti in una matrice. Dovete verificare tre aspetti:

- L'utente ha inserito numeri n^2 al posto di qualche n?
- Ciascuno dei numeri 1, 2, ..., n^2 è veramente presente una sola volta nell'input dell'utente?
- Quando i numeri vengono disposti nel riquadro, la somma delle righe, delle colonne e delle diagonali è uguale?

Se la dimensione dell'input è un quadrato, verificate che tutti i numeri compresi fra 1 e n^2 siano presenti. Poi calcolate le somme delle righe, delle colonne e delle diagonali.

Esercizio P11.16. Implementate il seguente algoritmo per costruire quadrati magici n per n; funziona solo se n è dispari. Mettete un 1 in mezzo all'ultima riga. Dopo aver messo k nella posizione (i, j), mettete $k + 1$ nel riquadro in basso a destra rispetto alle posizioni di k andando a capo sui bordi. Ma se il riquadro in basso a destra è già occupato o se vi trovate nell'angolo in basso a destra, dovete passare al riquadro subito sopra. Ecco il quadrato di 5×5 che si ottiene seguendo questo metodo:

11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

Scrivete un programma il cui input sia il numero n e il cui output sia il quadrato magico di ordine n se n è dispari.

Un numero intero è chiamato "perfetto" quando la somma dei suoi fattori, incluso 1 ma non se stesso, è pari a quel numero. Ad esempio 6 è un numero perfetto perché 6 = = 1 + 2 + 3. Si scriva un programma che visualizzi tutti i numeri perfetti, con i rispettivi fattori, inferiori od uguali ad un numero indicato dall'utente.

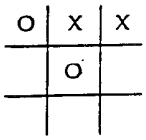
Esercizio P14.7. Scrivete un programma che simuli una macchina distributrice. I prodotti si possono acquistare inserendo nella macchina il numero giusto di monete. L'utente seleziona un prodotto da un elenco di prodotti disponibili, inserisce le monete e ottiene il prodotto oppure gli vengono restituite le monete se l'importo inserito non è sufficiente o se il prodotto è esaurito. Si può ripristinare la scorta dei prodotti e il denaro può essere ritirato da un operatore.

Si scriva un programma che definisca un calendario di appuntamenti. Ogni appuntamento è individuato da una data ed un'ora di inizio, da una data ed un'ora di fine e da una descrizione. Il programma, tramite un'opportuna interfaccia utente, consentirà di aggiungere ed eliminare appuntamenti, di scorrerli, nonché di visualizzare gli appuntamenti di una data giornata. Il programma consentirà inoltre di salvare l'elenco degli appuntamenti su un file e di leggere da file un elenco di appuntamenti precedentemente salvato. Aggiungere eventualmente al programma altre funzionalità, ad esempio la ricerca in base al campo descrizione.

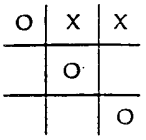
Esercizio P14.9. Posti in aereo. Scrivete un programma che assegni i posti su un aeroplano. Supponete che l'aeroplano abbia 20 posti in prima classe (5 file di 4 posti ciascuna, separate da un corridoio) e 180 posti in classe economica (30 file di 6 posti ciascuna, separate da un corridoio). Il vostro programma deve accettare tre comandi: aggiungere passeggeri, mostrare la distribuzione dei posti, uscire. Quando si aggiungono i passeggeri, dovete chiedere la classe (prima o economica), il numero di passeggeri che viaggiano assieme (1 o 2 in prima classe; da 1 a 3 in economica) e la posizione preferita (lato finestrino o corridoio in prima classe; corridoio, centro o finestrino in classe economica). Quindi dovrete tentare di trovare una corrispondenza e assegnare i posti. Se non esiste corrispondenza, stampate un messaggio. La vostra interfaccia utente potrà essere grafica o basata su testo.

Si scriva un programma che gestisca liste d'esami universitari. Il programma dovrà consentire all'utente di selezionare un esame, identificato da un codice, dalla materia e dalla data di effettuazione, e di inserire il nominativo dello studente, a sua volta identificato da matricola, nome e cognome, che desideri iscriversi all'esame stesso. Il programma dovrà inoltre consentire di salvare su file e recuperare da file la situazione relativa alle iscrizioni. Dovrà inoltre permettere le più comuni operazioni di gestione, quali ad esempio la cancellazione e la modifica dei dati inseriti.

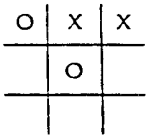
Esercizio P14.11. Scrivete un gioco del filetto ("tic-tac-toe") che consenta a un giocatore umano di giocare contro il computer. Il vostro programma giocherà molte partite contro un avversario umano e imparerà. Quando è il turno del computer, la macchina seleziona casualmente un campo libero, però non sceglierà mai una combinazione perdente. A questo scopo, il vostro programma deve tenere aggiornato un array di combinazioni perdenti. Tutte le volte che l'umano vince, la combinazione immediatamente precedente viene memorizzata come perdente. Per esempio, supponiamo che *x* sia il computer e *o* sia l'umano. Immaginiamo che la combinazione attuale sia:



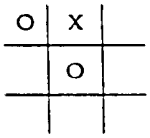
Ora è il turno dell'umano, che sceglierà naturalmente



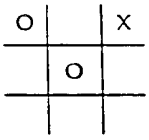
Il computer dovrebbe quindi ricordare la combinazione precedente



considerandola tale combinazione perdente. Come risultato, il computer non sceglierà mai un'altra volta una combinazione a partire da



oppure da



7.27 (*Il crivello di Eratostene*) Un numero primo è un intero divisibile soltanto per 1 e per sé stesso. Il Crivello di Eratostene è un metodo per trovare i numeri primi. Funziona così:

a) Create un array con tutti gli elementi inizializzati a 1 (`true`). Gli elementi dell'array che hanno per indice dei numeri primi resteranno uguali a 1. Tutti gli altri, invece, verranno impostati a zero.
b) Partite dall'indice 2 (l'indice 1 è necessariamente primo) e, ogni volta che trovate un elemento dell'array uguale a 1, impostate a zero tutti gli elementi il cui indice è multiplo dell'elemento trovato. Per l'elemento di indice 2, dovremo azzerare tutti gli elementi che si trovano dopo 2 e che sono multipli di due, cioè tutti gli elementi di indice pari; ugualmente, per l'elemento di indice 3, dovremo azzerare tutti gli elementi che hanno per indice un suo multiplo, come 6, 9, 12, 15, e così via.

Quando avremo completato il procedimento, gli elementi dell'array che valgono ancora 1 avranno per indice un numero primo. Scrivete un programma che implementi il crivello di Eratostene su 1000 elementi, per determinare i numeri primi compresi tra 1 e 999. Ignorate l'elemento 0 dell'array.

7.41 (*Simulazione: la lepre e la tartaruga*) In questo problema dovete ricreare la classica gara fra la lepre e la tartaruga. Utilizzerete i numeri casuali per la simulazione di questo evento.

I concorrenti iniziano la gara dal "riquadro 1", il primo di 70 riquadri. Ogni riquadro è una posizione possibile lungo il percorso della gara. La bandierina finale si trova sul riquadro 70. Il primo concorrente che raggiunge o sorpassa il riquadro 70 vince un succulento cestino pieno di carote e lattuga.

Il percorso si snoda lungo il versante in salita di una collinetta, per cui capita che uno dei due concorrenti possa perdere un po' di terreno. Un cronometro incrementa il suo valore di 1 tic al secondo. Ad ogni tic del cronometro il programma aggiorna la posizione degli animali, in base alle seguenti regole.

Animale	Mossa	% di tempo	Spostamento
Tartaruga	Passo veloce	50%	3 riquadri a destra
	Scivolata	20%	6 riquadri a sinistra
	Passo lento	30%	1 riquadro a destra
Lepre	Riposo	20%	Nessuno spostamento
	Salto lungo	20%	9 riquadri a destra
	Scivolata lunga	10%	12 riquadri a sinistra
	Salto piccolo	30%	1 riquadro a destra
	Scivolata piccola	20%	2 riquadri a sinistra

Utilizzate delle variabili per tenere traccia delle posizioni degli animali (ovvero del quadrato su cui si trovano, nell'intervallo 1-70). Fate partire ogni animale dalla posizione 1. Se un animale scivola ancora prima del riquadro 1, abbiate pietà e rimettetelo nuovamente sul riquadro 1.

Generate le percentuali della tabella precedente con un numero intero casuale i nell'intervallo $1 \leq i \leq 10$. Per la tartaruga, fatela andare a passo veloce quando $1 \leq i \leq 5$, fatela scivolare quando $6 \leq i \leq 7$, fatela andare a passo lento quando $8 \leq i \leq 10$. Usate una tecnica simile per spostare la lepre.

Ogni gara che si rispetti comincia con lo sparo iniziale. Visualizzate quindi

```
BANG !!!!!

PARTITI !!!!!
```

Per ogni tic del cronometro (ovvero a ogni iterazione del ciclo), visualizzate una linea con 70 tacche, dove mostrate la posizione della tartaruga con una `T` e quella della lepre con una `L`. Può anche capitare che i due concorrenti capitino sullo stesso riquadro. In questo caso, la tartaruga morde la lepre e il programma visualizzerà un `OUCH!!!` in quella posizione. Tutti gli spazi dove non c'è né la `T`, né la `L` e neppure `OUCH!!!` dovrebbero restare vuoti.

Dopo avere visualizzato ogni linea, controllate se uno degli animali ha raggiunto o sorpassato il riquadro 70. Se è così, visualizzate il nome del vincitore e terminate la simulazione. Se vince la tartaruga visualizzate `HA VINTO LA TARTARUGA!!! YAHOO!!!` Per par condicio, naturalmente, utilizzate una scritta simile anche se vince la lepre. E se entrambi raggiungono il riquadro 70 nello stesso momento? Fate come credete. Se volete potete favorire in ogni caso la tartaruga, oppure chiedere una rivincita. Se nessuno degli animali vince, effettuate il ciclo di nuovo per il successivo tic del cronometro.

Si supponga di avere un file di testo contenente un certo numero di parole di senso compiuto.
Si scriva un programma che scelga a caso una delle parole contenute nel file di testo e proponga all'utente di indovinare tale parola.
L'utente tenterà quindi di indovinare, una alla volta, le lettere contenute nella parola. Ogni volta che viene indovinata una lettera, questa verrà visualizzata nella posizione corretta all'interno della parola stessa. Se invece la lettera proposta dall'utente non appartiene alla parola, si avrà un errore. Sarà consentito all'utente di compiere soltanto un numero limitato di errori.

10.26 (*Codice Morse*) Il codice Morse assegna una serie di punti e trattini a ogni lettera dell'alfabeto, a ogni numero e ad alcuni caratteri speciali (come il punto, la virgola; i due punti e il punto e virgola). Nei sistemi basati sui suoni, il punto rappresenta un suono breve, mentre il trattino un suono più prolungato. La separazione delle parole è indicata da uno spazio o, più semplicemente, dall'assenza di un punto o di un trattino. Nei sistemi basati sui suoni, lo spazio è indicato da un breve periodo di tempo, durante il quale non viene trasmesso alcun suono. La figura 10.22 mostra la versione internazionale del Codice Morse.

Scrivete un'applicazione che legga una frase e la traduca in codice Morse. Scrivete inoltre un programma che legga una frase in codice Morse e la traduca in italiano. Usate uno spazio bianco tra ogni lettera del codice Morse e tre spazi bianchi tra ogni parola.

Morse Code

A •—	B —•••	C —•—•	D —••
E •	F ••—•	G ——•	H ••••
I ••	J •———	K —•—	L •—••
M ——	N —•	O ———	P •——•
Q ——•—	R •—•	S •••	T —
U ••—	V •••—	W •——	X —••—
Y —•——	Z ——••	0 —————	1 •————
2 ••———	3 •••——	4 ••••—	5 •••••
6 —••••	7 ——•••	8 ———••	9 ————•

Esercizio P13.5. *Cifrario Playfair.* Un altro modo per complicare la semplice analisi della frequenza delle lettere di un testo cifrato consiste nel cifrare le lettere *a coppie*. Il cifrario Playfair è uno schema semplice. Si prende una parola chiave e si eliminano le lettere doppie. Poi si inserisce la parola chiave, insieme con le lettere rimanenti dell'alfabeto, in un quadrato di 5 × 5. (Dato che ci sono solo 25 quadretti, I e J sono considerati la stessa lettera.) Ecco come si presenta questo tipo di disposizione con la parola chiave `PLAYFAIR`:

```
P  L  A  Y  F
I  R  B  C  D
E  G  H  K  M
N  O  Q  S  T
U  V  W  X  Z
```

Per cifrare una coppia di lettere, diciamo `AT`, osservate il rettangolo che ha per vertici `A` e `T`:

```
P  L  A  Y  F
I  R  B  C  D
E  G  H  K  M
N  O  Q  S  T
U  V  W  X  Z
```

Questa coppia si cifra cercando gli altri due vertici del rettangolo, in questo caso `FQ`. Se le due lettere si trovano sulla stessa riga, come per esempio `GO`, basta scambiarle. Per decifrare si usa lo stesso sistema.

Scrivete un programma per cifrare o decifrare un testo in base a questo cifrario.

Si realizzi un programma che simuli il gioco del lotto.
Il programma consentirà all'utente di simulare la puntata su uno o più numeri di una o più ruote.
Alla pressione di un pulsante il programma effettuerà l'estrazione casuale dei numeri sulle varie ruote e calcolerà l'eventuale vincita dell'utente.
Si inseriscano inoltre nel programma alcune semplici funzionalità statistiche.

NB: Le regole del gioco sono semplici e facilmente reperibili in Internet.

Si realizzi un programma che simuli il famoso gioco televisivo “Chi vuol essere milionario”.

Il gioco prevede che il concorrente (cioè, nel caso del programma, l’utente) risponda in sequenza ad una serie di domande, incrementando progressivamente il montepremi.
Per ogni domanda al concorrente vengono proposte quattro possibili risposte tra cui scegliere, una sola delle quali esatta.
Se il concorrente sceglie la risposta esatta, passa alla domanda successiva, e così in successione fino alla domanda finale con il montepremi più elevato. In caso di errore invece il gioco ha termine ed il concorrente è eliminato.

Il programma dovrà essere in grado di leggere le domande e le corrispondenti possibili risposte da file di testo e di presentare le stesse in modo casuale. Inoltre, le domande dovranno essere in qualche modo classificate in base alla difficoltà, in modo che a montepremi più elevati corrispondano domande più difficili.

Regole più dettagliate sul gioco, dalle quali trarre spunto per aggiungere ulteriori funzionalità al programma, sono facilmente reperibili in Internet.