Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 23 febbraio 2018

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su TUTTI i fogli.*

1. Considerate funzione $f : \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

k	$f(k)$
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7

k	$f(k)$
j	7
k	7
l	8
m	9
n	10
o	10
p	11
q	12
r	12

k	$f(k)$
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

Sia h la funzione che trasforma ogni parola k sull'alfabeto $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ nell'intero che si ottiene applicando f al primo carattere di k e g la funzione che trasforma ogni parola k nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando f *al terzo* carattere di k.

Esempi: $h(\mathtt{gatto}) = 6$, $g(\mathtt{gatto}) = 13$, $h(\mathtt{cozza}) = 2$, $g(\mathtt{cozza}) = 17$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

`topo tigre struzzo lepre lupo cavallo cane leone`

Come funzione hash utilizzate h. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`9 4 7 15 12 10 11`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una scacchiera $n \times n$ partendo da una casella della colonna più a sinistra. Il robot è alimentato con delle mele, collocate in un apposito serbatoio.

- Inizialmente nel serbatoio vengono collocate tutte le mele che si trovano nella casella della colonna di sinistra da cui il robot inizia a muoversi.
- Per spostarsi da una casella il robot ha bisogno di mangiare K mele (ciò avviene anche quando il robot si sposta da una casella per abbandonare la scacchiera). Se il robot ha meno di K mele rimane bloccato in attesa di soccorsi.
- Ogni volta che il robot raggiunge una casella, mette nel serbatoio tutte le mele che vi trova (per semplicità supponete che la capacità del serbatoio sia illimitata).

Il robot può scegliere arbitrariamente una casella della colonna di sinistra da cui iniziare il proprio percorso. Ad ogni spostamento il robot può muoversi esclusivamente in una delle seguenti caselle della colonna immediatamente a destra: la casella sulla stessa riga, la casella sulla riga superiore (se esiste), la casella sulla riga inferiore (se esiste). L'obiettivo è abbandonare la scacchiera spostandosi all'esterno da una qualunque casella della colonna di destra.

Esempio. La figura qui sotto a sinistra rappresenta una scacchiera con l'indicazione del numero di mele presenti in ciascuna casella. Nelle altre figure sono evidenziati quattro possibili cammini nel caso $K = 5$. Nei primi due cammini il robot riesce ad uscire a destra, negli altri rimane bloccato.

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

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

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

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

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

Cosa si richiede. Sono dati una matrice M di dimensione $n \times n$, contenente interi non negativi (l'elemento $m_{i,j}$ è il numero di mele presenti nella casella (i, j)) e il valore K (numero di mele necessarie per spostarsi da una casella a un'altra). Indichiamo con $c_{i,j}$ il massimo numero di mele che il robot può avere nel serbatoio seguendo un percorso che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j), assegnamo a $c_{i,j}$ il valore $-\infty$. Si osservi che $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$. Sia C la matrice dei valori $c_{i,j}$.

Esempio. Dati $M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 8 & 2 & 0 & 3 \\ 0 & 6 & 4 & 1 \\ 5 & 7 & 1 & 5 \end{bmatrix}$ e $K = 5$ (esempio precedente), si ottiene $C = \begin{bmatrix} 8 & 6 & 5 & 4 \\ 8 & 5 & 4 & 6 \\ 0 & 9 & 8 & 4 \\ 5 & 7 & 5 & 8 \end{bmatrix}$. Si noti che in questo caso esiste almeno un percorso che permette al robot di uscire dalla scacchiera.

Per la stessa matrice M, con $K = 6$ si ottiene $C = \begin{bmatrix} 8 & 5 & -\infty & -\infty \\ 8 & 4 & 2 & 3 \\ 0 & 8 & 6 & 1 \\ 5 & -\infty & 3 & 5 \end{bmatrix}$. In questo caso non esiste alcun percorso che permette al robot di uscire dalla scacchiera.

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule o, in alternativa, righe di pseudocodice, che permettano di ricavare il generico valore $c_{i,j}$ della colonna j di C, con $j > 1$, sulla base di K, di alcuni valori della colonna $j - 1$ di C e del valore $m_{i,j}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M e il valore K riempia la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo da una casella nella colonna di sinistra di uscire dalla colonna di destra.

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

Note

- Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x.
- Al punto (b) si richiede solo di stabilire *se esiste un cammino* (è sufficiente che l'algoritmo risponda "sì" oppure "no"). Non è richiesto di ricavare il cammino.
- Non cambiate i nomi stabiliti nel testo dell'esercizio, cioè i nomi delle matrici e della costante K.

Cognome . . . . . .

Nome . . . . . .

Matricola . . . . . .

Algoritmi e Strutture Dati

Prova scritta del 23 febbraio 2018

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su TUTTI i fogli.*

1. Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

k	$f(k)$
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7

k	$f(k)$
j	7
k	7
l	8
m	9
n	10
o	10
p	11
q	12
r	12

k	$f(k)$
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

Sia h la funzione che trasforma ogni parola k sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando f al primo carattere di k e g la funzione che trasforma ogni parola k nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando f *al terzo* carattere di k.

Esempi: $h(\texttt{gatto}) = 6$, $g(\texttt{gatto}) = 13$, $h(\texttt{cozza}) = 2$, $g(\texttt{cozza}) = 17$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
tarantola sogliola struzzo lupo lepre coyote
coniglio lince
```

Come funzione hash utilizzate h. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`10 5 8 16 13 11 12`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una scacchiera $n \times n$ partendo da una casella della colonna più a sinistra. Il robot è alimentato con delle mele, collocate in un apposito serbatoio.

- Inizialmente nel serbatoio vengono collocate tutte le mele che si trovano nella casella della colonna di sinistra da cui il robot inizia a muoversi.
- Per spostarsi da una casella il robot ha bisogno di mangiare K mele (ciò avviene anche quando il robot si sposta da una casella per abbandonare la scacchiera). Se il robot ha meno di K mele rimane bloccato in attesa di soccorsi.
- Ogni volta che il robot raggiunge una casella, mette nel serbatoio tutte le mele che vi trova (per semplicità supponete che la capacità del serbatoio sia illimitata).

Il robot può scegliere arbitrariamente una casella della colonna di sinistra da cui iniziare il proprio percorso. Ad ogni spostamento il robot può muoversi esclusivamente in una delle seguenti caselle della colonna immediatamente a destra: la casella sulla stessa riga, la casella sulla riga superiore (se esiste), la casella sulla riga inferiore (se esiste). L'obiettivo è abbandonare la scacchiera spostandosi all'esterno da una qualunque casella della colonna di destra.

Esempio. La figura qui sotto a sinistra rappresenta una scacchiera con l'indicazione del numero di mele presenti in ciascuna casella. Nelle altre figure sono evidenziati quattro possibili cammini nel caso $K = 5$. Nei primi due cammini il robot riesce ad uscire a destra, negli altri rimane bloccato.

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

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

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

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

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

Cosa si richiede. Sono dati una matrice M di dimensione $n \times n$, contenente interi non negativi (l'elemento $m_{i,j}$ è il numero di mele presenti nella casella (i, j)) e il valore K (numero di mele necessarie per spostarsi da una casella a un'altra). Indichiamo con $c_{i,j}$ il massimo numero di mele che il robot può avere nel serbatoio seguendo un percorso che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j), assegnamo a $c_{i,j}$ il valore $-\infty$. Si osservi che $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$. Sia C la matrice dei valori $c_{i,j}$.

Esempio. Dati $M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 8 & 2 & 0 & 3 \\ 0 & 6 & 4 & 1 \\ 5 & 7 & 1 & 5 \end{bmatrix}$ e $K = 5$ (esempio precedente), si ottiene $C = \begin{bmatrix} 8 & 6 & 5 & 4 \\ 8 & 5 & 4 & 6 \\ 0 & 9 & 8 & 4 \\ 5 & 7 & 5 & 8 \end{bmatrix}$. Si noti che in questo caso esiste almeno un percorso che permette al robot di uscire dalla scacchiera.

Per la stessa matrice M, con $K = 6$ si ottiene $C = \begin{bmatrix} 8 & 5 & -\infty & -\infty \\ 8 & 4 & 2 & 3 \\ 0 & 8 & 6 & 1 \\ 5 & -\infty & 3 & 5 \end{bmatrix}$. In questo caso non esiste alcun percorso che permette al robot di uscire dalla scacchiera.

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule o, in alternativa, righe di pseudocodice, che permettano di ricavare il generico valore $c_{i,j}$ della colonna j di C, con $j > 1$, sulla base di K, di alcuni valori della colonna $j - 1$ di C e del valore $m_{i,j}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M e il valore K riempia la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo da una casella nella colonna di sinistra di uscire dalla colonna di destra.

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

Note

- Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x.
- Al punto (b) si richiede solo di stabilire *se esiste un cammino* (è sufficiente che l'algoritmo risponda "sì" oppure "no"). Non è richiesto di ricavare il cammino.
- Non cambiate i nomi stabiliti nel testo dell'esercizio, cioè i nomi delle matrici e della costante K.

Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 23 febbraio 2018

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su TUTTI i fogli.*

1. Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

k	$f(k)$
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7

k	$f(k)$
j	7
k	7
l	8
m	9
n	10
o	10
p	11
q	12
r	12

k	$f(k)$
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

Sia h la funzione che trasforma ogni parola k sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando f al primo carattere di k e g la funzione che trasforma ogni parola k nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando f *al terzo* carattere di k.

Esempi: $h(\texttt{gatto}) = 6$, $g(\texttt{gatto}) = 13$, $h(\texttt{cozza}) = 2$, $g(\texttt{cozza}) = 17$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
trota tigre tartaruga lepre lupo corvo condor
leone
```

Come funzione hash utilizzate h. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`11 6 9 17 14 12 13`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una scacchiera $n \times n$ partendo da una casella della colonna più a sinistra. Il robot è alimentato con delle mele, collocate in un apposito serbatoio.

- Inizialmente nel serbatoio vengono collocate tutte le mele che si trovano nella casella della colonna di sinistra da cui il robot inizia a muoversi.
- Per spostarsi da una casella il robot ha bisogno di mangiare K mele (ciò avviene anche quando il robot si sposta da una casella per abbandonare la scacchiera). Se il robot ha meno di K mele rimane bloccato in attesa di soccorsi.
- Ogni volta che il robot raggiunge una casella, mette nel serbatoio tutte le mele che vi trova (per semplicità supponete che la capacità del serbatoio sia illimitata).

Il robot può scegliere arbitrariamente una casella della colonna di sinistra da cui iniziare il proprio percorso. Ad ogni spostamento il robot può muoversi esclusivamente in una delle seguenti caselle della colonna immediatamente a destra: la casella sulla stessa riga, la casella sulla riga superiore (se esiste), la casella sulla riga inferiore (se esiste). L'obiettivo è abbandonare la scacchiera spostandosi all'esterno da una qualunque casella della colonna di destra.

Esempio. La figura qui sotto a sinistra rappresenta una scacchiera con l'indicazione del numero di mele presenti in ciascuna casella. Nelle altre figure sono evidenziati quattro possibili cammini nel caso $K = 5$. Nei primi due cammini il robot riesce ad uscire a destra, negli altri rimane bloccato.

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

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

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

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

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

Cosa si richiede. Sono dati una matrice M di dimensione $n \times n$, contenente interi non negativi (l'elemento $m_{i,j}$ è il numero di mele presenti nella casella (i, j)) e il valore K (numero di mele necessarie per spostarsi da una casella a un'altra). Indichiamo con $c_{i,j}$ il massimo numero di mele che il robot può avere nel serbatoio seguendo un percorso che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j), assegnamo a $c_{i,j}$ il valore $-\infty$. Si osservi che $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$. Sia C la matrice dei valori $c_{i,j}$.

Esempio. Dati $M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 8 & 2 & 0 & 3 \\ 0 & 6 & 4 & 1 \\ 5 & 7 & 1 & 5 \end{bmatrix}$ e $K = 5$ (esempio precedente), si ottiene $C = \begin{bmatrix} 8 & 6 & 5 & 4 \\ 8 & 5 & 4 & 6 \\ 0 & 9 & 8 & 4 \\ 5 & 7 & 5 & 8 \end{bmatrix}$. Si noti che in questo caso esiste almeno un percorso che permette al robot di uscire dalla scacchiera.

Per la stessa matrice M, con $K = 6$ si ottiene $C = \begin{bmatrix} 8 & 5 & -\infty & -\infty \\ 8 & 4 & 2 & 3 \\ 0 & 8 & 6 & 1 \\ 5 & -\infty & 3 & 5 \end{bmatrix}$. In questo caso non esiste alcun percorso che permette al robot di uscire dalla scacchiera.

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule o, in alternativa, righe di pseudocodice, che permettano di ricavare il generico valore $c_{i,j}$ della colonna j di C, con $j > 1$, sulla base di K, di alcuni valori della colonna $j - 1$ di C e del valore $m_{i,j}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M e il valore K riempia la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo da una casella nella colonna di sinistra di uscire dalla colonna di destra.

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

Note

- Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x.
- Al punto (b) si richiede solo di stabilire *se esiste un cammino* (è sufficiente che l'algoritmo risponda "sì" oppure "no"). Non è richiesto di ricavare il cammino.
- Non cambiate i nomi stabiliti nel testo dell'esercizio, cioè i nomi delle matrici e della costante K.

Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 23 febbraio 2018

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su TUTTI i fogli.*

1. Considerate funzione $f : \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\} \to \{0, 1, \ldots, 15\}$ definita come segue:

k	$f(k)$
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7

k	$f(k)$
j	7
k	7
l	8
m	9
n	10
o	10
p	11
q	12
r	12

k	$f(k)$
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

Sia h la funzione che trasforma ogni parola k sull'alfabeto $\{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ nell'intero che si ottiene applicando f al primo carattere di k e g la funzione che trasforma ogni parola k nel *più piccolo numero primo* maggiore o uguale al valore che si ottiene applicando f *al terzo* carattere di k.

Esempi: $h(\texttt{gatto}) = 6$, $g(\texttt{gatto}) = 13$, $h(\texttt{cozza}) = 2$, $g(\texttt{cozza}) = 17$.

Inserite nella tabella hash a destra, inizialmente vuota, le seguenti parole, nell'ordine indicato:

```
tonno sogliola tartaruga lupo lepre cobra canarino
lince
```

Come funzione hash utilizzate h. Per la gestione delle collisioni utilizzate l'hashing doppio mediante la funzione

$$c(k, i) = (\, h(k) + i \cdot g(k) \,) \bmod 16$$

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`12 7 10 18 15 13 14`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

3. Un robot si muove su una scacchiera $n \times n$ partendo da una casella della colonna più a sinistra. Il robot è alimentato con delle mele, collocate in un apposito serbatoio.

- Inizialmente nel serbatoio vengono collocate tutte le mele che si trovano nella casella della colonna di sinistra da cui il robot inizia a muoversi.
- Per spostarsi da una casella il robot ha bisogno di mangiare K mele (ciò avviene anche quando il robot si sposta da una casella per abbandonare la scacchiera). Se il robot ha meno di K mele rimane bloccato in attesa di soccorsi.
- Ogni volta che il robot raggiunge una casella, mette nel serbatoio tutte le mele che vi trova (per semplicità supponete che la capacità del serbatoio sia illimitata).

Il robot può scegliere arbitrariamente una casella della colonna di sinistra da cui iniziare il proprio percorso. Ad ogni spostamento il robot può muoversi esclusivamente in una delle seguenti caselle della colonna immediatamente a destra: la casella sulla stessa riga, la casella sulla riga superiore (se esiste), la casella sulla riga inferiore (se esiste). L'obiettivo è abbandonare la scacchiera spostandosi all'esterno da una qualunque casella della colonna di destra.

Esempio. La figura qui sotto a sinistra rappresenta una scacchiera con l'indicazione del numero di mele presenti in ciascuna casella. Nelle altre figure sono evidenziati quattro possibili cammini nel caso $K = 5$. Nei primi due cammini il robot riesce ad uscire a destra, negli altri rimane bloccato.

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

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

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

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

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

Cosa si richiede. Sono dati una matrice M di dimensione $n \times n$, contenente interi non negativi (l'elemento $m_{i,j}$ è il numero di mele presenti nella casella (i, j)) e il valore K (numero di mele necessarie per spostarsi da una casella a un'altra). Indichiamo con $c_{i,j}$ il massimo numero di mele che il robot può avere nel serbatoio seguendo un percorso che inizia in una qualunque casella della colonna di sinistra (cioè la numero 1) e termina nella casella (i, j). Nel caso il robot non sia in grado di raggiungere la casella (i, j), assegnamo a $c_{i,j}$ il valore $-\infty$. Si osservi che $c_{i,1} = m_{i,1}$, per $i = 1, \ldots, n$. Sia C la matrice dei valori $c_{i,j}$.

Esempio. Dati $M = \begin{bmatrix} 8 & 3 & 4 & 4 \\ 8 & 2 & 0 & 3 \\ 0 & 6 & 4 & 1 \\ 5 & 7 & 1 & 5 \end{bmatrix}$ e $K = 5$ (esempio precedente), si ottiene $C = \begin{bmatrix} 8 & 6 & 5 & 4 \\ 8 & 5 & 4 & 6 \\ 0 & 9 & 8 & 4 \\ 5 & 7 & 5 & 8 \end{bmatrix}$. Si noti che in questo caso esiste almeno un percorso che permette al robot di uscire dalla scacchiera.

Per la stessa matrice M, con $K = 6$ si ottiene $C = \begin{bmatrix} 8 & 5 & -\infty & -\infty \\ 8 & 4 & 2 & 3 \\ 0 & 8 & 6 & 1 \\ 5 & -\infty & 3 & 5 \end{bmatrix}$. In questo caso non esiste alcun percorso che permette al robot di uscire dalla scacchiera.

Risolvete i seguenti punti ***nell'ordine indicato.***

(a) Tenendo conto delle regole con cui il robot si può muovere, scrivete una o più formule o, in alternativa, righe di pseudocodice, che permettano di ricavare il generico valore $c_{i,j}$ della colonna j di C, con $j > 1$, sulla base di K, di alcuni valori della colonna $j - 1$ di C e del valore $m_{i,j}$.

(b) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice, un algoritmo basato sulla tecnica di *programmazione dinamica* che ricevendo in ingresso la matrice M e il valore K riempia la matrice C (utilizzate quanto scritto al punto (a)) e stabilisca se esista almeno un percorso che permetta al robot partendo da una casella nella colonna di sinistra di uscire dalla colonna di destra.

(c) Fornite una stima del tempo totale utilizzato dall'algoritmo in funzione di n.

Note

- Potete effettuare operazioni direttamente con il valore $-\infty$. Il risultato dell'addizione e della sottrazione tra $-\infty$ e un valore finito è $-\infty$. Il minimo e il massimo tra $-\infty$ e un valore x sono rispettivamente $-\infty$ e x.
- Al punto (b) si richiede solo di stabilire *se esiste un cammino* (è sufficiente che l'algoritmo risponda "sì" oppure "no"). Non è richiesto di ricavare il cammino.
- Non cambiate i nomi stabiliti nel testo dell'esercizio, cioè i nomi delle matrici e della costante K.