

## Esercizi svolti di Informatica A - parte II (rappresentazione dell'informazione)

### a) Rappresentazione del testo

Per questo esercizio considerate la seguente porzione di tabella ASCII:

<b>Carattere:</b>	a	b	c	d	e	f	g	h	i	j	k	l	m
<b>Decimale:</b>	97	98	99	100	101	102	103	104	105	106	107	108	109
<b>Hex:</b>	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D
<b>Carattere:</b>	n	o	p	q	r	s	t	u	v	w	x	y	z
<b>Decimale:</b>	110	111	112	113	114	115	116	117	118	119	120	121	122
<b>Hex:</b>	6E	6F	70	71	72	73	74	75	76	77	78	79	7A

Scrivete la rappresentazione ASCII (**usando i codici esadecimali**) delle seguenti parole, completando la tabella:

#### Soluzione

è sufficiente riportare per ogni singola lettera il corrispondente codice esadecimale!

parola	rappresentazione ASCII
banana	62616E616E61
esadecimale	657361646563696D616C65
informatica	696E666F726D6174696361
pippo	706970706F
supercalifragilistichepiralidoso	737570657263616C6966726167696C69737469636865 73706972616C69646F736F
facebook	66616365626F6F6B

Ora procedete a rovescio, ossia decodificate le seguenti parole rappresentate coi codici ASCII esadecimali:

#### Soluzione











In questo caso è utile suddividere la rappresentazione ASCII a coppie di simboli: ogni coppia è infatti il codice esadecimale di una ben precisa lettera. Fatto questo possiamo andare a cercare le corrispondenze con le lettere nella tabella qui sopra.

rappresentazione ASCII	parola
65 73 61 6D 65	esame
6C 65 69 62 6E 69 7A	leibniz
6D 61 67 69 63 61 62 75 6C 61	magicabula

65 64 75 63 61 7A 69 6F 6E 65	educazione
73 63 69 65 6E 7A 61	scienza
71 75 69 64 64 69 74 63 68	quidditch

## b) Rappresentazione delle immagini

Negli esercizi ci concentreremo sulla rappresentazione RGB. Ricordiamo quello che serve sapere per svolgere l'esercizio:

- ogni pixel è rappresentato da 3 componenti, che rappresentano rispettivamente i colori ROSSO, VERDE e BLU, nell'ordine.
- ogni componente è formata da un byte (ossia 8 bit, o 2 cifre esadecimali: come nelle dispense di Start@Unito useremo quest'ultima rappresentazione) e dice "quanto" di quel colore va a comporre il colore finale
- per semplicità nei nostri esercizi una componente è o totalmente assente (codice esadecimale 00) o totalmente presente (codice esadecimale FF) o è presente in "metà dose" (codice esadecimale 80)
- i tre colori fondamentali ( ROSSO,  VERDE e  BLU) hanno – ovviamente – solo una componente totalmente presente: quella che gli corrisponde
- ecco come si ottengono i colori che useremo negli esercizi
  -  BIANCO: tutte le componenti **presenti**
  -  NERO: tutte le componenti **assenti**
  -  GRIGIO: tutte le componenti a **metà**
  -  GIALLO: ROSSO + VERDE
  -  ARANCIONE: ROSSO + metà dose VERDE
  -  MAGENTA: ROSSO + BLU
  -  CIANO (azzurro): VERDE + BLU

Provare a indicare la rappresentazione RGB coi codici esadecimali delle seguenti immagini (per ciascuna immagine, la versione ingrandita permette meglio di vedere il colore di ciascun pixel, mentre quella più piccola permette di cogliere meglio il senso dell'immagine stessa).

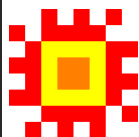
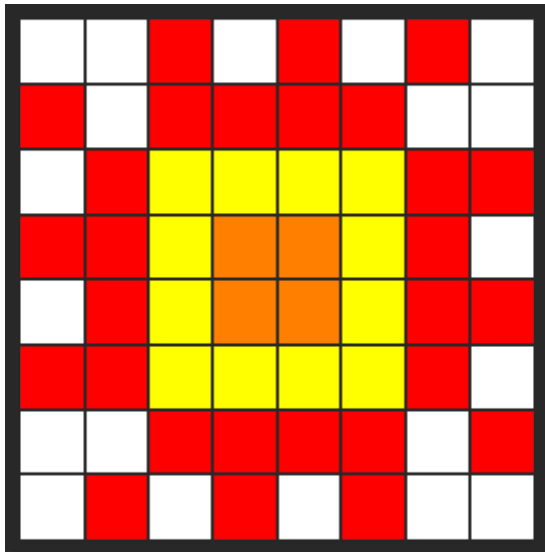
(suggerimento: ogni immagine è composta da  $8 \times 8 = 64$  pixel; ogni pixel è rappresentato con 6 cifre esadecimali – 2 per componente di colore – quindi ci aspettiamo una sequenza di  $64 \times 6 = 384$  cifre esadecimali.)

### Soluzione:

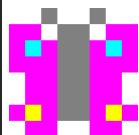
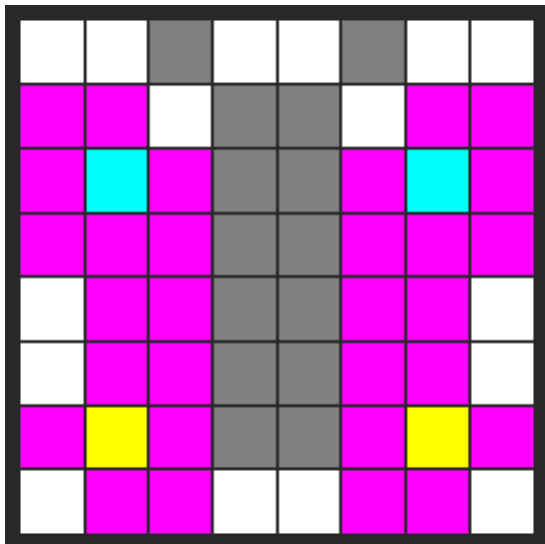
Una volta ricevute indicazioni come quelle qui sopra, il primo passo è costruire una tabella coi colori rilevanti, riportando i corrispondenti codici esadecimali in formato RGB. Negli esercizi d'esame ci saranno meno colori e immagini con meno pixel!

									
rosso	verde	bianco	nero	grigio	giallo	arancione	magenta	azzurro	blu
FF0000	00FF00	FFFFFF	000000	808080	FFFF00	FF8000	FF00FF	00FFFF	0000FF

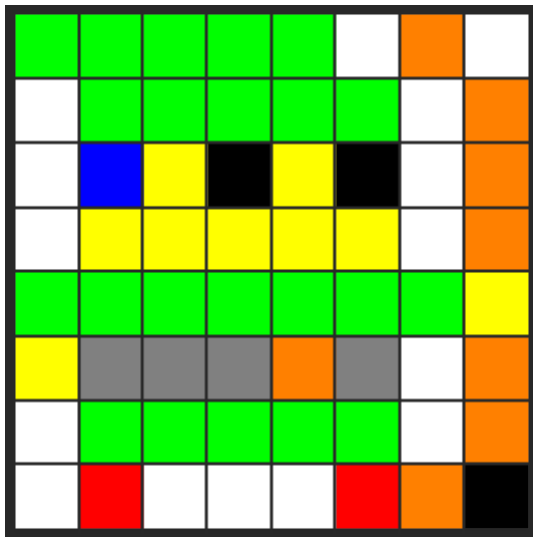
Una volta ottenuta questa tabella, sarà sufficiente elencare per ogni pixel il codice del colore corrispondente. Per semplicità separiamo i codici con uno spazio, e andiamo a capo alla fine di ogni riga dell'immagine.



```
FFFFFF FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF
FF0000 FFFFFFFF FF0000 FF0000 FF0000 FF0000 FFFFFFFF FFFFFFFF
FFFFFF FF0000 FFFF00 FFFF00 FFFF00 FFFF00 FF0000 FF0000
FF0000 FF0000 FFFF00 FF8000 FF8000 FFFF00 FF0000 FFFFFFFF
FFFFFF FF0000 FFFF00 FF8000 FF8000 FFFF00 FF0000 FF0000
FF0000 FF0000 FFFF00 FFFF00 FFFF00 FFFF00 FF0000 FFFFFFFF
FFFFFF FFFFFFFF FF0000 FF0000 FF0000 FF0000 FFFFFFFF FF0000
FFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF FF0000 FFFFFFFF FFFFFFFF
```



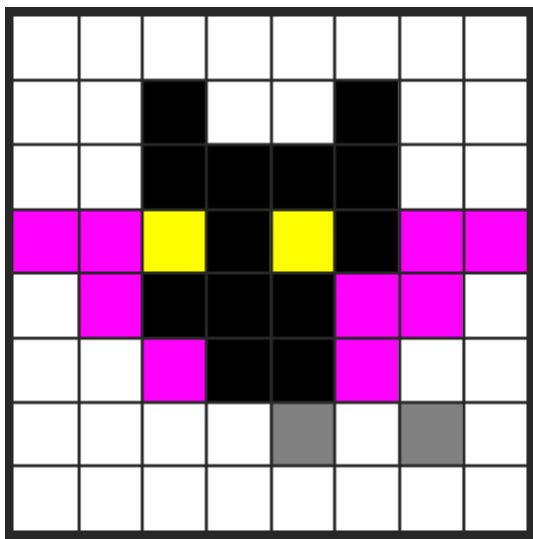
```
FFFFFF FFFFFFFF 808080 FFFFFFFF FFFFFFFF 808080 FFFFFFFF FFFFFFFF
FF00FF FF00FF FFFFFFFF 808080 808080 FFFFFFFF FF00FF FF00FF
FF00FF 00FFFF FF00FF 808080 808080 FF00FF 00FFFF FF00FF
FF00FF FF00FF FF00FF 808080 808080 FF00FF FF00FF FF00FF
FFFFFF FF00FF FF00FF 808080 808080 FF00FF FF00FF FFFFFFFF
FFFFFF FF00FF FF00FF 808080 808080 FF00FF FF00FF FFFFFFFF
FF00FF FFFF00 FF00FF 808080 808080 FF00FF FFFF00 FF00FF
FFFFFF FF00FF FF00FF FFFFFFFF FFFFFFFF FF00FF FF00FF FFFFFFFF
```



```

00FF00 00FF00 00FF00 00FF00 00FF00 FFFFFFFF FF8000 FFFFFFFF
FFFFFF 00FF00 00FF00 00FF00 00FF00 00FF00 FFFFFFFF FF8000
FFFFFF 0000FF FFFF00 000000 FFFF00 000000 FFFFFFFF FF8000
FFFFFF FFFF00 FFFF00 FFFF00 FFFF00 FFFF00 FFFFFFFF FF8000
00FF00 00FF00 00FF00 00FF00 00FF00 00FF00 00FF00 FFFF00
FFFF00 808080 808080 808080 FF8000 808080 FFFFFFFF FF8000
FFFFFF 00FF00 00FF00 00FF00 00FF00 00FF00 FFFFFFFF FF8000
FFFFFF FF0000 FFFFFFFF FFFFFFFF FFFFFFFF FF0000 FF8000 000000

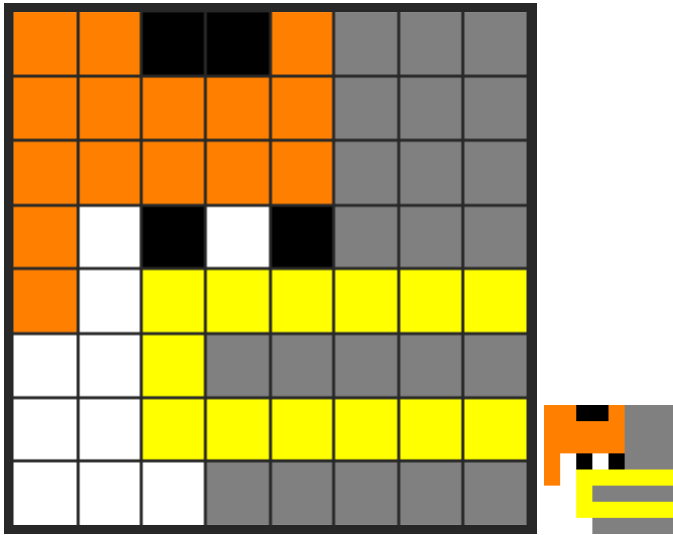
```



```

FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFF FFFFFFFF 000000 FFFFFFFF FFFFFFFF 000000 FFFFFFFF FFFFFFFF
FFFFFF FFFFFFFF 000000 000000 000000 000000 FFFFFFFF FFFFFFFF
FF00FF FF00FF FFFF00 000000 FFFF00 000000 FF00FF FF00FF
FFFFFF FF00FF 000000 000000 000000 FF00FF FF00FF FFFFFFFF
FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 808080 FFFFFFFF 808080 FFFFFFFF
FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

```



FF8000	FF8000	000000	000000	FF8000	808080	808080	808080
FF8000	FF8000	FF8000	FF8000	FF8000	808080	808080	808080
FF8000	FF8000	FF8000	FF8000	FF8000	808080	808080	808080
FF8000	FFFFFF	000000	FFFFFF	000000	808080	808080	808080
FF8000	FFFFFF	FFFF00	FFFF00	FFFF00	FFFF00	FFFF00	FFFF00
FFFFFF	FFFFFF	FFFF00	808080	808080	808080	808080	808080
FFFFFF	FFFFFF	FFFF00	FFFF00	FFFF00	FFFF00	FFFF00	FFFF00
FFFFFF	FFFFFF	FFFFFF	808080	808080	808080	808080	808080