Esercizio 1 (punti 6)

Nel file `array.c` implementare la definizione della funzione:

```
extern int *crea_inizializza (unsigned int n, int val);
```

La funzione accetta come parametri la dimensione di un vettore di int `n` e un valore intero `val` e deve restituire un puntatore ad un vettore allocato dinamicamente nell'heap, formato da `n` elementi in cui il primo vale `val`, il secondo `val-1` e così via. Ad esempio, dati n=5 e val=3, il vettore conterrà i valori 3, 2, 1, 0, -1.

Esercizio 2 (punti 6)

Creare i file `alterna.h` e `alterna.c` che consentano di utilizzare la seguente funzione:

```
extern char *alterna (const char *s1, const char *s2);
```

La funzione accetta come parametri due stringhe zero terminate e deve restituire un puntatore ad una nuova stringa zero terminata (allocata dinamicamente nell'heap) formata dall'alternarsi dei caratteri della prima e della seconda stringa. Se le stringhe non hanno la stessa lunghezza, la nuova stringa terminerà con i caratteri rimasti della stringa più lunga. Ad esempio, date le stringhe "prova" e "1234567" la funzione deve ritornare la stringa "p1r2o3v4a567".

Esercizio 3 (punti 6)

Nel file `cross.c` implementare la definizione della funzione:

```
extern void stampa_cross (unsigned int n);
```

La funzione deve inviare a `stdout` una X composta da un carattere 'x' al centro e caratteri '\' e '/' sulle diagonali. Ogni semi-diagonale deve essere composta di `n` caratteri. Ad esempio chiamando la funzione con `n=0`, la funzione deve inviare su stdout solo il centro:

```
x
```

Chiamando la funzione con `n=2`, la funzione deve inviare su stdout:

```
\   /
 \ /
  x
 / \
/   \
```

Ovvero (visualizzando ogni carattere in una cella della seguente tabella):

\	<spazio>	<spazio>	<spazio>	/	<a capo>
<spazio>	\	<spazio>	/	<spazio>	<a capo>
<spazio>	<spazio>	x	<spazio>	<spazio>	<a capo>
<spazio>	/	<spazio>	\	<spazio>	<a capo>
/	<spazio>	<spazio>	<spazio>	\	<a capo>

Esercizio 4 (punti 8)

Creare i file `libri.h` e `libri.c` che consentano di utilizzare la seguente struttura:

```
struct libro {
	unsigned int codice;
	char titolo[255];
	unsigned int pagine;
};
```

e la funzione:

```
extern struct libro *read_libri (const char *filename, unsigned int *pn);
```

La struttura contiene il campo `codice` che contiene un identificatore di un libro, il campo `titolo` che ne contiene il titolo (che può includere degli spazi) e il campo `pagine` che contiene il numero di pagine.
La funzione accetta come parametro un nome di file che deve essere aperto in lettura in modalità tradotta (testo) e un puntatore ad una variabile di tipo `unsigned int` in cui si dovrà inserire il numero di libri presenti in un file così strutturato:

```
<codice>;<titolo>;<pagine><a capo>
<codice>;<titolo>;<pagine><a capo>
<codice>;<titolo>;<pagine><a capo>
...
```

La funzione deve ritornare un puntatore ad una nuova zona di memoria (allocata dinamicamente nell'heap) contenente tutti i libri letti dal file. Il numero di libri non è noto a priori e non può essere vincolato dal codice.

Ad esempio, un file valido è:

```
23875;L'uomo che piantava gli alberi (Salani Ragazzi);64
75628;Programmazione C. Le basi per tutti (Esperto in un click);111
76890;L'arte dell'hacking - volume 1 (Pocket);336
12374;Piccolo manuale della sicurezza informatica (Pocket);204
```

In questo caso la funzione dovrà impostare la variabile puntata da `pn` a 4. Per testare la funzione, utilizzare i file `libri1.txt` e `libri2.txt` disponibili nella pagina dell'esame.

Esercizio 5 (punti 7)

Creare i file `libri.h` e `libri.c` che consentano di utilizzare la seguente struttura:

```
struct libro {
	unsigned int codice;
	char titolo[255];
	unsigned int pagine;
};
```

e la funzione:

```
struct libro *filtra_libri (struct libro *plibri, unsigned int *pn, const char *cerca);
```

La struttura contiene il campo `codice` che contiene un identificatore di un libro, il campo `titolo` che ne contiene il titolo (che può includere degli spazi) e il campo `pagine` che contiene il numero di pagine.
La funzione accetta come parametro `plibri`, un puntatore ad una zona di memoria contenente un numero di libri indicato dal valore puntato da `pn`. Accetta inoltre una stringa zero terminata `cerca`. La funzione deve ritornare un puntatore ad una nuova zona di memoria (allocata dinamicamente nell'heap) contenente tutti i libri nel cui titolo è presente la stringa contenuta in `cerca` e impostare il valore puntato da `pn` al numero di libri ritornati. Se nessun libro contiene la stringa, la funzione deve ritornare `NULL` e impostare il valore puntato da `pn` a `0`.

Ad esempio, se `plibri` contenesse i seguenti dati:

23875	L'uomo che piantava gli alberi (Salani Ragazzi)	64
75628	Programmazione C. Le basi per tutti (Esperto in un click)	111
76890	L'arte dell'hacking - volume 1 (Pocket)	336
12374	Piccolo manuale della sicurezza informatica (Pocket)	204

e `pn` puntasse al valore 4, chiamando la funzione con `cerca="del"`, la funzione deve ritornare un puntatore contenente:

76890	L'arte dell'hacking - volume 1 (Pocket)	336
12374	Piccolo manuale della sicurezza informatica (Pocket)	204

e impostare la variabile puntata da `pn` a 2.