Esercizio 1

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

```
size_t conta_spazi (const char* s);
```

La funzione accetta come parametro una stringa zero terminata e deve restituire il numero di caratteri <spazio> presenti nella stessa.
Ad esempio, data la stringa “prova stringa in cui contare gli spazi” la funzione deve ritornare il valore 6.

Esercizio 2

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

```
extern char *concatena(const char *prima, const char *seconda);
```

La funzione riceve due puntatori a stringhe di caratteri zero terminate e alloca dinamicamente sull’heap sufficiente memoria per contenerle entrambe (incluso un terminatore) e copia nel nuovo spazio allocato la prima, seguita dalla seconda. Un esempio di chiamata è il seguente:

```
int main(void) {
	char s1[] = "Questa e' la ";
	char s2[] = "stringa risultante.";
	char *s;

	s = concatena(s1, s2);

	free(s);
}
```

In questo caso s punterà ad un vettore di caratteri contenente "Questa e’ la stringa risultante.". Se uno dei puntatori (`prima` o `seconda`) è NULL o punta ad una stringa vuota (cioè il primo carattere vale 0), la funzione lo tratta come una stringa di lunghezza 0. Ad esempio chiamando `concatena("a","")`, si allocano 2 char e la stringa ritornata contiene il carattere `'a'` e il carattere 0.

Esercizio 3

Sia data la struct seguente:

```
struct punto {
	double x, y;
};
```

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

```
extern int colineari(struct punto p1, struct punto p2, struct punto p3);
```

La funzione ritorna 1 se p_1, p_2 e p_3 giacciono sulla stessa retta, altrimenti 0. L'equazione da verificare è la seguente:

$$(x_3 - x_2)(y_1 - y_2) = (y_3 - y_2)(x_1 - x_2)$$

Esercizio 4

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

```
struct complesso {
	double re,im;
};
```

e la funzione:

```
extern void prodotto_complesso (struct complesso *comp1, const struct complesso *comp2);
```

La funzione `prodotto_complesso` esegue il prodotto dei due valori `comp1` e `comp2` e mette il risultato in `comp1`. Si ricorda che il prodotto di numeri complessi si esegue così:

$$(a + ib)(c + id) = (ac - bd) + i(ad + bc)$$

Esercizio 5

Creare il file `encrypt.c` che contenga la definizione della seguente funzione:

```
extern void encrypt(char *s, unsigned int n);
```

La funzione accetta una sequenza `s` di `n` char e la codifica sostituendo ad ogni char il suo valore trasformato con uno XOR bit a bit con il valore esadecimale AA. Per le proprietà dello XOR, l'operazione è invertibile, quindi, riapplicando la funzione sulla sequenza codificata, si riottiene quella originale. Non è richiesto, né è garantito che la sequenza sia 0 terminata. Per questo motivo viene passato un parametro apposito.