Aufgabe 1a)

Teil 1:

`x` nimmt am Ende den Wert -8 an, weil mit `x *= *y*x;` folgendes bedeutet:

- `x *=` : multipliziere x, welches durch `int x = -2` den Wert -2 hat mit:
- `*y`: y an sich speichert durch `*y = &x;` die Adresse, in der die Daten für x gespeichert sind. Durch `*y` wird der Zeiger dereferenziert, d.h. er gibt den Wert zurück, der in der Adresse gespeichert ist, die y gespeichert hat. Da y die Adresse von x speichert, ist `*y` gleich -2
- `*y*x` : das `*x` ist keine Dereferenzierung, sondern Multiplikation
- Mit eingesetzten Werten bedeutet der Term `-2 *= -2*-2;` was -8 ergibt

Teil 2:

`x` nimmt am Ende wieder den Wert 10 an.

- `*p = &x` ist ein Zeiger, der die Adresse des Wertes x speichert
- `*q = p` ist ein Zeiger, der die Adresse speichert, die gerade in `*p` gespeichert ist
- `(*p)++` durch den Stern wird p dereferenziert, es wird also auf den Wert zugegriffen, den p speichert, also x: x wird 11 durch `++`
- `--(*q)` durch den Stern wird die in q gespeicherte Adresse dereferenziert, es wird also somit die Adresse dereferenziert, die in p gespeichert ist, welche die Variable x ist
    - x wird `-(*q)` um 1 erniedrigt und ist wieder 10

Teil 3:

`x` nimmt am Ende den Wert 6 an.

- `*p` ist wieder ein Zeiger auf x, `**p2` speichert die Adresse des Zeigers p1, der auf x zeigt
- `(*p2 > p1)`: p2 wurde nur einmal dereferenziert, d.h. er gibt die Adresse des Zeigers p1 zurück
- Dadurch, dass `p1 == *p2` ist, wird die Aussage `*p2 > p1` nie wahr sein, weshalb
- `++(**p2)` ausgeführt wird: p2 wird zweimal dereferenziert, beim ersten Mal gibt er die Adresse von p1 zurück, durch die zweite Dereferenzierung wird auf x zugegriffen, da p1 auf x zeigt
- durch das ++ wird x um 1 erhöht

Aufgabe 1b)

Das Problem ist in Zeile 2: man versucht p1 und p2 als Zeiger auf a und b zu definieren. Das funktioniert nur bei p1, da der Stern nur p1 zum Pointer macht. Um auch p2 zum Pointer zu machen, müsste die Zeile lauten: `int *p1 = &a, *p2 = &b;`.