1 String-Konkatenation

Implementieren Sie `string_concat` im folgenden Programm so, dass die beiden übergebenen Strings `string1` und `string2` zusammenhängend in den String `out` geschrieben werden. Benutzen Sie für die eigentliche Konkatenation keine von der Standardbibliothek bereitgestellten Funktionen wie z. B. `sprintf`. Testen Sie auch Spezialfälle wie leere Strings.

Erinnerung: Ein String ist ein Array von `char`-Werten, das mit einem Null-Byte (`\0`) terminiert ist.

```c
#include <stdio.h>

void string_concat (char out[], char string1[], char string2[])
{
}

int main (void)
{
    char out[128] = { '\0' };
    string_concat(out, "Hello", "World");
    printf("%s\n", out);
    return 0;
}
```

2 Strukturen und Unions

Modifizieren Sie das folgende Programm so, dass auf oberster Ebene nur noch ein `struct foobar` deklariert wird, das die beiden Strukturen `struct foo` und `struct bar` enthält. Um den Speicherplatz gering zu halten, soll außerdem eine Union eingesetzt werden, so dass `struct foobar` maximal 8 Bytes groß ist.

```c
#include <stdio.h>

struct foo
{
    int a;
    int b;
};

struct bar
{
    float a;
    float b;
};

```

```c
int main (void)
{
    struct foo a = { 42, 23 };
    printf("foo: %d %d\n", a.a, a.b);
    struct bar b = { 23.0, 42.0 };
    printf("bar: %f %f\n", b.a, b.b);
    return 0;
}
```

2.1 Enums

Passen Sie das gegebene Programm weiterhin so an, dass ein `enum` genutzt wird, um in `struct foobar` zu speichern, ob die Integer- oder die Float-Komponente aktiv ist. Die Größe von `struct foobar` darf sich dadurch erhöhen.