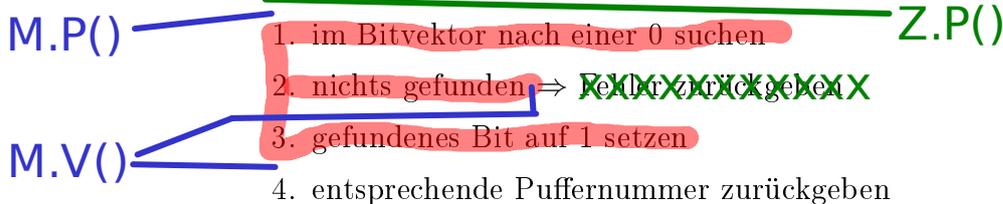


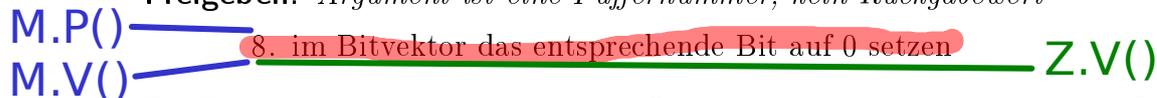
Aufgabe V6: Semaphore im Einsatz

Eine Pufferverwaltung auf Anwendungsebene erlaubt es, Puffer fester Größe anzufordern und wieder freizugeben. Die Anzahl der Puffer wird einmalig bei der Initialisierung festgelegt. Die Puffer sind ab 0 durchnummeriert. Die Verwaltung verwendet einen Bitvektor entsprechender Länge, in dem 0 einen freien und 1 einen belegten Puffer anzeigt. Die Operationen laufen wie folgt ab:

Anfordern: *kein Argument, Rückgabewert ist eine Puffernummer oder ein Fehler*



Freigeben: *Argument ist eine Puffernummer, kein Rückgabewert*



Die Pufferverwaltung ist noch nicht für parallele Aufrufe durch mehrere Prozesse (engl.: *threads*) gleichzeitig ausgelegt. Gehen Sie davon aus, dass die Anwendung nur gültige Aufrufe durchführt, also nur zuvor angeforderte Puffer freigibt.

- Bestimmen Sie die kritischen Abschnitte beider Operationen.
- Sichern Sie die Pufferverwaltung mit einem Semaphore M so ab, dass **höchstens ein kritischer Abschnitt gleichzeitig** ablaufen kann. Mit welchem Wert wird M initialisiert? Wo müssen Aufrufe von $M.P()$ bzw. $M.V()$ stattfinden?

- Ändern Sie die Pufferverwaltung mit einem zweiten Semaphore Z so, dass das Anfordern auf einen freien Puffer wartet, statt einen Fehler zurückzugeben. Mit welchem Wert wird Z initialisiert? Wo müssen Aufrufe von $Z.P()$ bzw. $Z.V()$ stattfinden?
- Was kann schiefgehen, wenn zwei Prozesse gleichzeitig die ungesicherte Pufferverwaltung nutzen, also ohne das Semaphore M ?
 - beide rufen **Anfordern**
 - beide rufen **Freigeben**
 - einer ruft **Anfordern**, der andere **Freigeben**