Anmerkungen zur Abgabe der Hausübung 2

- Generell wird der Rückgabewert `0` erwartet, falls der Aufruf von `njvm` erfolgreich war.
- Alle Binärdateien der Tests haben die Versionsnummer 8. Achten Sie also darauf, dass die Abgabeversion Ihrer `njvm` auch die Version 8 der Binärdateien akzeptiert.
- Einige Tests prüfen, ob der Garbage Collector (GC) nicht genügend Speicher bereitstellen kann. In diesem Fall muss der Returnwert `1` sein.
- Einige Tests prüfen, ob die VM mehr Speicher anfordert als erlaubt. In diesem Fall muss der Returnwert `1` sein.
- Denken Sie daran, dass Sie auch die `bigint`-Bibliothek kompilieren müssen.
    - Angenommen die `bigint`-Bibliothek befindet sich im Verzeichnis `src/bigint`, dann wäre der Aufruf
      `cd src/bigint;make;cd ../..` — Anschließend befindet man sich wieder im Ausgangsverzeichnis.

Beispiel: Kompilation der bigint-Bibliothek

```
ar@vmar01:[~/KSP_public_WS20_21/hausuebung/njvm]$ ls
 mknjvm src
ar@vmar01:[~/KSP_public_WS20_21/hausuebung/njvm]$ ls src/
bigint  debug.c  debug.h  func.c  func.h  global.c  global.h  memory.c  memory.h
njvm.c  readNJBF.c  readNJBF.h  stackop.c  stackop.h  support.c
ar@vmar01:[~/KSP_public_WS20_21/hausuebung/njvm]$ cat mknjvm
#!/usr/bin/env bash

cd src/bigint/; make; cd ../..
gcc -g -Wall -pedantic -std=c99 -Isrc/bigint/build/include -Lsrc/bigint/build/lib -o
njvm src/debug.c src/func.c src/global.c src/memory.c src/njvm.c src/readNJBF.c
src/stackop.c src/support.c -lbigint
ar@vmar01:[~/KSP_public_WS20_21/hausuebung/njvm]$ ./mknjvm
for i in src tst ; do \
          make -C $i install ; \
        done
make[1]: Entering directory '/home/ar/njvm/src/bigint/src'
mkdir -p ../build/include
cp support.h ../build/include
cp bigint.h ../build/include
mkdir -p ../build/lib
cp libbigint.a ../build/lib
make[1]: Leaving directory '/home/ar/njvm/src/bigint/src'
make[1]: Entering directory '/home/ar/njvm/src/bigint/tst'
mkdir -p ../build/bin
cp testbip ../build/bin
make[1]: Leaving directory '/home/ar/njvm/src/bigint/tst'
ar@vmar01:[~/KSP_public_WS20_21/hausuebung/njvm]$ ls
 mknjvm   njvm  'README(Submit).txt'   src   submit.sh
```

- Das gegebene Programm `factor.nj` aus Aufgabe 8, ist einer der umfangreichsten Tests. Wenn dieses Programm bei Ihnen ohne Probleme läuft, dann werden Sie mit hoher Wahrscheinlichkeit auch die HU2 bestehen.
    - Bevor Sie den GC implementieren stellen Sie sicher, dass das Programm `factor.nj` bei Ihnen exakt die gleiche Ausgabe erzeugt wie in der Referenzimplementierung.
    - Treten nach dem Implementieren des GCs dann Fehler auf, wissen Sie es liegt an der Implementierung vom GC selbst.
- Der Returnwert `139` bedeutet eigentlich in allen Fällen, dass Ihr Programm einen sog. `segmentation fault` ausgelöst hat. In den überwiegenden Fällen passiert dies, wenn man eine *ungültige* Speicheradresse dereferenziert (z.B. `NULL`).
    - Prüfen Sie immer darauf, ob `malloc` bzw. Ihre eigene Allokationsfunktion Ihnen eine gültige Speicheradresse zurückgibt (`!=NULL`)!
- Beachten Sie in jedem Fall die neuen Programmparameter `--stack` und `--heap` aus Aufgabe 8. Fast alle Testfälle verwenden diese Parameter mit unterschiedlichen Größenangaben. In jedem Fall sollte Ihre `njvm` die eigene Speicherreservierung aus Aufgabe 8 implementieren, und nur so

viel Speicher anfordern wie mittels den Parametern angegeben.

- Die Implementierung des GC ist **optional** und für das Bestehen der HU2 nicht unbedingt nötig. Um volle Punktzahl bei den Tests zu erhalten, muss der GC natürlich vollständig implementiert sein!
    - Achtung ohne die Implementierung des GC müssen nahezu alle anderen Tests ohne irgendwelche Fehler laufen, ansonsten kann es bei der Abgabe eng werden!