17/20 Eine der besten Abgabe, tolle Arbeit! und schön aufgeschrieben, eine

6+1 weil ihr eine der wenigen Gruppen seid die es korrekt gemacht haben = 7 /6 perfekt

BS Blatt06

Aufgabe 1

code:

```
// Deklaration x
SUBI SP 1                 # x auf Stack laden
LOADI ACC 15
STOREIN SP ACC 1
LOADIN SP ACC 1           # Wert vom Stack holen und Stack erhöhen (freigeben)
ADDI SP 1
STORE ACC 128             # x-Wert bei bds = 128 speichern

// Deklaration y
SUBI SP 1                 # y auf Stack laden
LOADI ACC 3
STOREIN SP ACC 1
LOADIN SP ACC 1           # Wert vom Stack holen und Stack erhöhen (freigeben)
ADDI SP 1
STORE ACC 129             # y-Wert bei bds=129 speichern

// Werte für Schleife auf Stack laden
SUBI SP 1                 # x auf Stack
LOAD ACC 128
STOREIN SP ACC 1
```

```
SUBI SP 1                   # y auf Stack
LOAD ACC 129
STOREIN SP ACC 1

SUBI SP 1                   # z auf Stack
LOADi ACC 2
STOREIN SP ACC 1

//z*y berechnen
LOADIN SP ACC 2             # y-Wert in ACC
LOADIN SP IN2 1             # z-Wert in IN2
MUL ACC IN2                 # z * y berechnen
STOREIN SP ACC 2            # Ergebnis auf Stack speichern und Stack erhöhen
                            (freigeben)
ADDI SP 1

// Schleifenbedingung
LOADIN SP ACC 2             # x-Wert in ACC
LOADIN SP IN2 1             # y*z-Ergebnis in IN2
SUB ACC IN2                 # statt x <= z * y ----> x - (z*y) >= 0
JUMP>= 3                    # Falls Ergebnis >= 0 --> Ergebnis 1 ansonsten Ergebnis 0
LOADI ACC 0
JUMP 2
LOADI ACC 1
STOREIN SP ACC 2            # Erg. der Operation vom ACC in Stack und dann Platz freig.
ADDI SP 1
```

```
// Schleife
LOADIN SP ACC 1          # Erg. von <= in ACC und prüfen, wenn 0 dann
                         schleifeninhalt ansonsten nicht
ADDI SP 1
JUMP== 16

// Schleifeninhalt
SUBI SP 1                # x-Wert in Stack laden
LOAD ACC 128
STOREIN SP ACC 1
SUBI SP 1                # 3 in Stack laden
LOADI ACC 3
STOREIN SP ACC 1

LOADIN SP ACC 2          # x-Wert in Acc und 3 in IN2
LOADIN SP IN2 1
SUB ACC IN2              # x = x -3 berechnen und erg in Stack speichern + freigabe
STOREIN SP ACC 2
ADDI SP 1

LOADIN SP ACC 1          # Neuer x-Wert an bds = 128 speichern
STORE ACC 128
ADDI SP 1
JUMP -40
```

hab nicht genau nachgezählt, aber falls mein Pfeil dahinzeigt wo ihr das dachtet, dann ist es korrekt

aufgabe 3

a = &(p2.x); → a zeigt auf Adresse 10. Auf diese wird Adresse 15 geschrieben.

p2.x = 7; → Auf Adresse 15 wird 7 geschrieben
p2.y = 4; → Auf Adresse 16 wird 4 geschrieben

p1 = (struct point *) malloc(sizeof(struct point)); → Adresse 33 wird auf Adresse 8 gespeichert

(*p1).y = *a; → a, welche auf Adresse 10 zeigt ist die Adresse 15 mit dem Wert 7. (*p1).y, welcher auf Adresse 33 verweist (+ 1 wegen size(y)) also auf Adresse 34 kommt der Wert 7

p3 = p1; → p3 zeigt auf Adresse 9. Auf diese wird Adresse 33 geschrieben.
p1 = &p2; → p1 zeigt auf Adresse 8 und da wird Adresse 15 geschrieben

if((*p1).y > 5) → Auf Adresse 15 +1 (wegen size(y)) also 16 wird der Wert 4 ausgelesen und mit 5 verglichen

*a = 42; → Auf a auf Adresse 10 wo die Adresse 15 steht wird der Wert 42 also auf Adresse 15 geschrieben. Wird aber durch die Abfrage nicht ausgeführt und geht ins else

*a = 1; → Auf a auf Adresse 10 wo wie Adresse 15 steht wird der Wert 42 also auf Adresse 15 geschrieben.

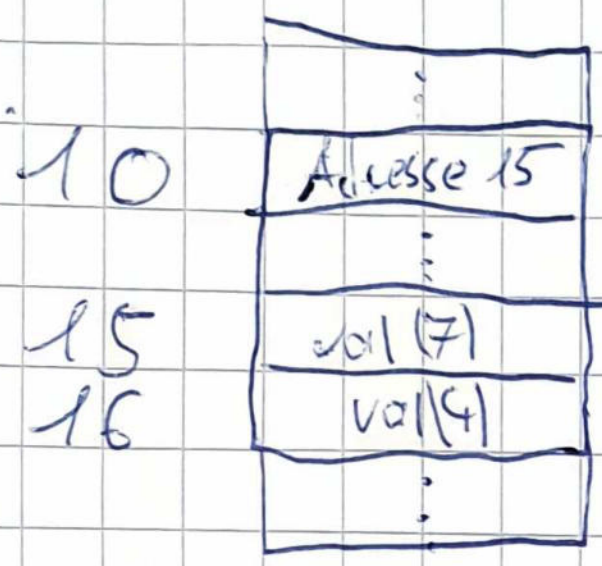
free(p3) → Adresse 33 wird freigegeben bzw. der Heap.

10/10

die b) beantwortet ja iwie auch schon direkt die a)

9/9

b) Marke 1



Marke 2

Marke 3

c) Da über p3 noch auf den Heap gezeigt wird, wird der Heap freigegeben (33 und 34)

1/1