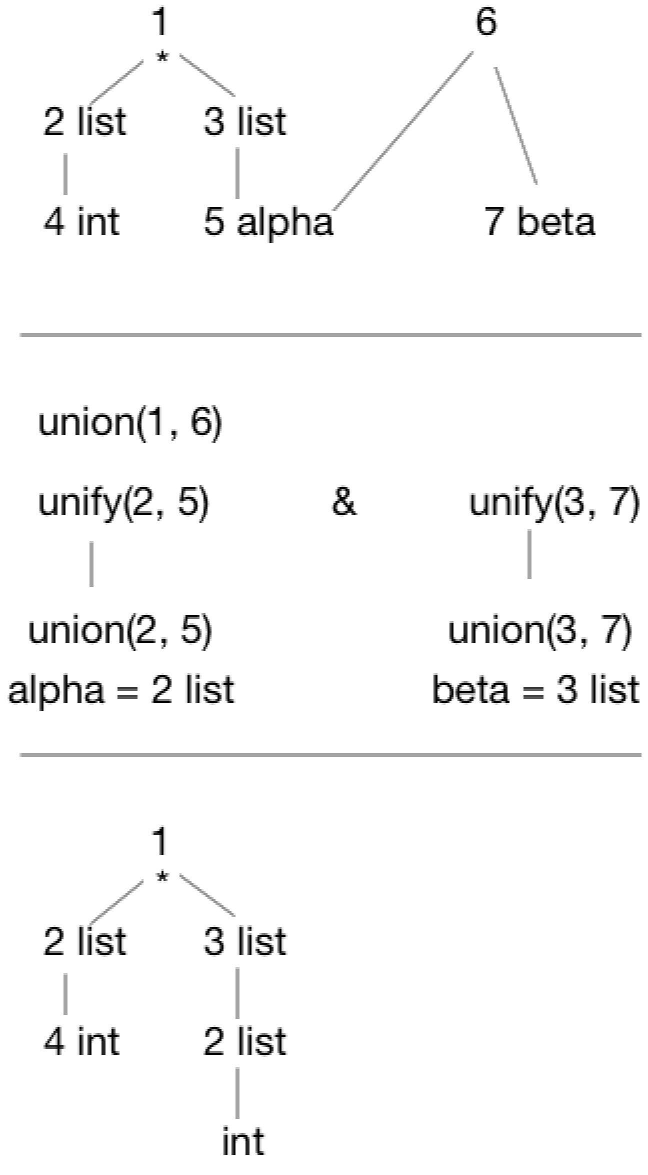
Ov assign 3

Task 1



resultatet er list int * list list int

Task 2

- a)

```
t_1 := v + w
t_2 := w + 1
LABEL _GCD_
  IF t_1 = 0 THEN
    t_0 := t_2 * 2
  ELSE
    t_1 := t_2
    t_2 := t_1 mod t_2
    GOTO _GCD_
```

- b)

IL

```
t_1 := v
t_2 := w
LABEL loop
  IF t_2 == 0 THEN
    GOTO exit
  IF  t_1 / t_2 != 0 THEN
    IF t_1 < t_2 THEN
      t_1 := t_1 - t_2
    ELSE
      t_2 := t_2 - t_1
    GOTO loop

LABEL exit
```

ASM

```
main:
  j cond

lf:
  subi $s1, $s2
  j cond

loop:
slt $t1, $s2, $s1
bneq $t1, $zero, lf
subi $s2, $s1
```

```
cond:
  beq $s2, $zero, exit
  div $t1, $s1, $s2
  bneq $t1, $zero, loop

exit:
```

- c)

```
seq $s0 $s2, $s3
xori $s1 $zero, $s4
```

Task 3

- a)

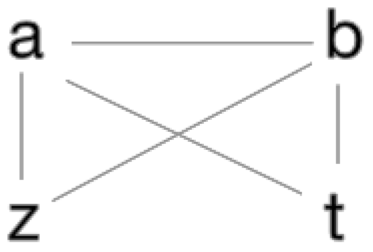
Instruction i	**gen[i]**	**kill[i]**	**succ[i]**
LABEL start	Ø	Ø	i + 1
IF a < b THEN next ELSE swap	{a, b}	Ø	{next, swap}
LABEL swap	Ø	Ø	i + 1
t := a	{a}	{t}	i + 1
a := b	{b}	{a}	i + 1
b := t	{t}	{b}	i + 1
LABEL next	Ø	Ø	i + 1
z := 0	Ø	{z}	i + 1
b := b mod a	{b, a}	{b}	i + 1
IF b = z THEN end ELSE start	{b, z}	Ø	{end, start}
LABEL end	Ø	Ø	i + 1
RETURN a	{a}	Ø	i + 1

- b)

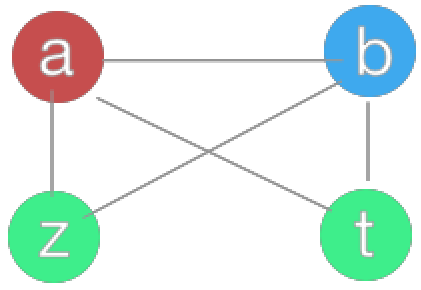
gen[i]	**kill[i]**	**in[i]**	**out[i]**	**in[i]**	**out[i]**	**in[i]**	**out[i]**	**in[i]**	**out[i]**
Ø	Ø	a,b	a,b	a,b	a,b	a,b	a,b	a,b	a,b
{a, b}	Ø	a,b	a	a,b	a	a,b	a,b	a,b	a,b
Ø	Ø	a	a	a	a,b	a,b	a,b	a,b	a,b
{a}	{t}	a	b,t	a,t	b,t	a,t	b,t	a,t	b,t
{b}	{a}	b,t	a,t	b,t	a,t	b,t	a,t	b,t	a,t
{t}	{b}	a,t	b,a	a,t	a,b	a,t	a,b	a,t	a,b
Ø	Ø	b,a	b,a	a,b	a,b	a,b	a,b	a,b	a,b
Ø	{z}	b,a	b,z,a	a, b	b,z,a	a,b	b,z,a	a,b	b,z,a
{a, b}	{b}	b,z,a	b,za	b,z,a	b,z,a	b,z,a	b,z,a	b,z,a	b,z,a
{b,z}	Ø	b,z,a	a	b,z,a	a	b,z,a	a	b,z,a	a, b
Ø	Ø	a	a	a	a	a	a	a	a
{a}	Ø	a		a		a		a	

- c)

out[i]	**kill[i]**	**LHS**	**Interference**
b,t	{t}	t	b
a,t	{a}	a	t
a,b	{b}	b	a
a,b,z	{z}	z	a,b
a,b,z	{b}	b	a,z



- d)



- e)

Knude	**Naboer**	**Farve**
b		blå
t	b	grøn
a	b,t	spild
z	a,b	grøn

```
M[a_addr] = a
LABEL start:
  a1 = M[a_addr]
  IF a1 < b THEN next ELSE swap

LABEL swap:
  a2 = M[a_addr]
  t := a2
  a3 := b
  M[a_addr] = a3
  b := t
```

```
LABEL next:
  z := 0
  a4 = M[a_addr]
  b := b mod a4
  IF b = z THEN end ELSE start

LABEL end:
  a = M[a_addr]
  RETURN a
```

Task 4

- a)

```
char *y = (char*)malloc(n);
int l, i = 0;

while (i++ < n) {
  if (f(*x++)) {
    ++*y = *x
    l++;
  }
}

y[0] = l
```

- b)

Se filter.asm (Det er compiled code, så nok lidt ulæseligt)

- c)

Resultatet bliver et int array og inputs skal være et int array.