Implementering af programmeringssprog - Skriftlig 4 timer

14

22 juni 2022

Planlagt: 17:00 - 21:00

Eksamensnr: 14

Plads: EH-2151

Opgave 1.

1.1

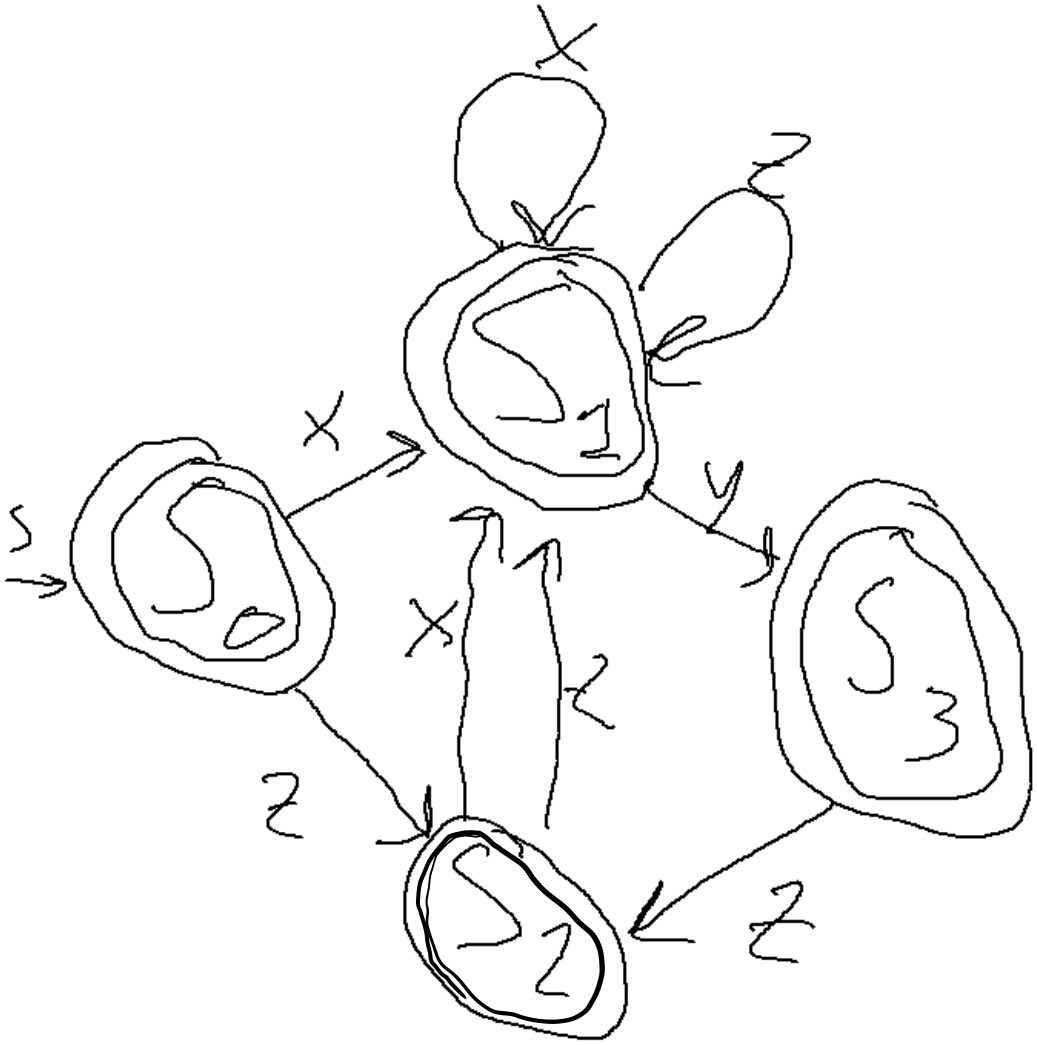
$move(s, \epsilon) = \epsilon - \{1\} = \{1,4\} = s_0$ accepting

$move(s_0, x) = \epsilon - \{2,3\} = \{1,2,3,4\} = s_1$ accepting
$move(s_0, y) = \epsilon - \{\quad\} = \{\quad\}$
$move(s_0, z) = \epsilon - \{2\} = \{1,2,4\} = s_2$ accepting

$move(s_1, x) = \epsilon - \{2,3\} = s_1$
$move(s_1, y) = \epsilon - \{4\} = \{4\} = s_3$ accepting
$move(s_1, z) = \epsilon - \{2,3\} = s_1$

$move(s_2, x) = \epsilon - \{2,3\} = s_1$
$move(s_2, y) = \epsilon - \{\quad\} = \{\}$
$move(s_2, z) = \epsilon - \{2,3\} = s_1$

$move(s_3, x) = \epsilon - \{\quad\} = \{\quad\}$
$move(s_3, y) = \epsilon - \{\quad\} = \{\quad\}$
$move(s_3, z) = \epsilon - \{2\} = s_2$



1.2

$$G_1 = \{2,5\}$$

~~$G_2 = \{1,3,4,6,7,D\}$~~

G_2	a	b
1	G_2	G_2
3	G_1	G_2
4	G_2	G_1
6	G_2	G_2
7	G_2	G_2

D	G_2	G_2

G_2 bliver lavet om til

$$\sout{G_2 = 1,6,7,D}$$

$$G_4 = 3$$

$$G_5 = 4$$

G_3	a	b
1	G_5	G_4
6	G_3	G_3
7	G_3	G_3
D	G_3	G_3

G_3bliver lavet om til

$$G_6 = 1$$

$$G_7 = 6,7,D$$

G_7	a	b
6	G_7	G_7
7	G_7	G_7
D	G_7	G_7

G_1	a	b
2	G_1	G_6
5	G_1	G_6

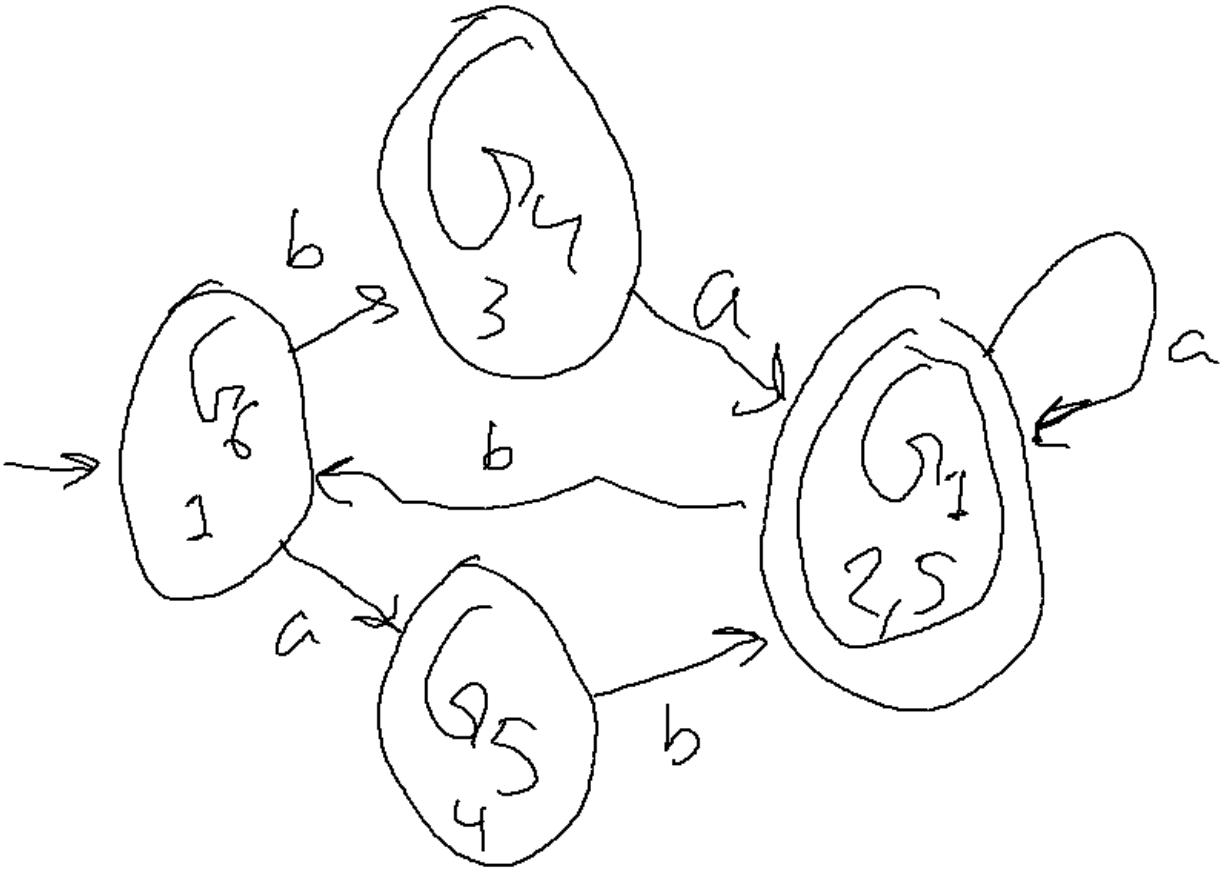
G_4	a	b
3	G_1	G_7

G_5	a	b
4	G_7	G_1

G_6	a	b

1	G_5	G_4

G_7 er blevet fjernet, da det er en død state.



1.3

a) Det er muligt, at lave en RegEx, da man kan lave en, for alle keywords, og trække dem fra den man har lavet for identifiers, som starter med en bogstav, efterfulgt at bogstaver, tal og underscore.

b) Det ville være muligt, da man kan lave en regex, som kan have alle tegn (inklusiv intet tegn) foran sig, og det samme efter sig, og så med et givent keyword i midten, eller flere keywords i midten.

[0-127]*(int | in | ..|) [0-127]*

hvor [0-127] er alle ascii tegn

c) Igen kan man have alle tegn, inklusiv intet tegn, foran, og det samme efter sig, med 2 keywords i midten.

[0-127]*(intint | inin| .. |)[0-127]*

hvor [0-127] er alle ascii tegn

d) Det kan man ikke, hvis der er ikke er en begrænsning på, hvor langed identifierne kan være, da den givene regex vil blive uendeligt stort. Hvis der er en grænse, kan man matche op mod alt, inden for den givene længde, men det vil blive en meget stor regex.

e) Man kan have alle tegn, inklusiv intet tegn, foran, og det samme efter sig, med et keyword, efterfulgt af alle tegn, inklusiv intet tegn efter sig, efterfulgt af det samme keyword igen.

[0-127]*(int [0-127]* int | in [0-127]* in | ..|) [0-127]*

hvor [0-127] er alle ascii tegn

Opgave 2.

2.1

a)

S

A

aA

aaP

aaA +

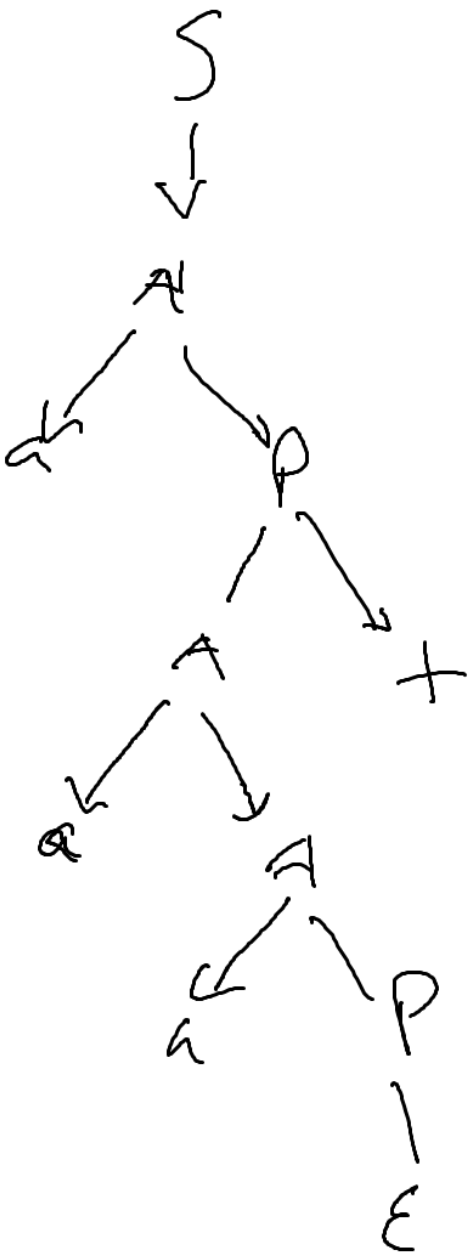
aaaP+

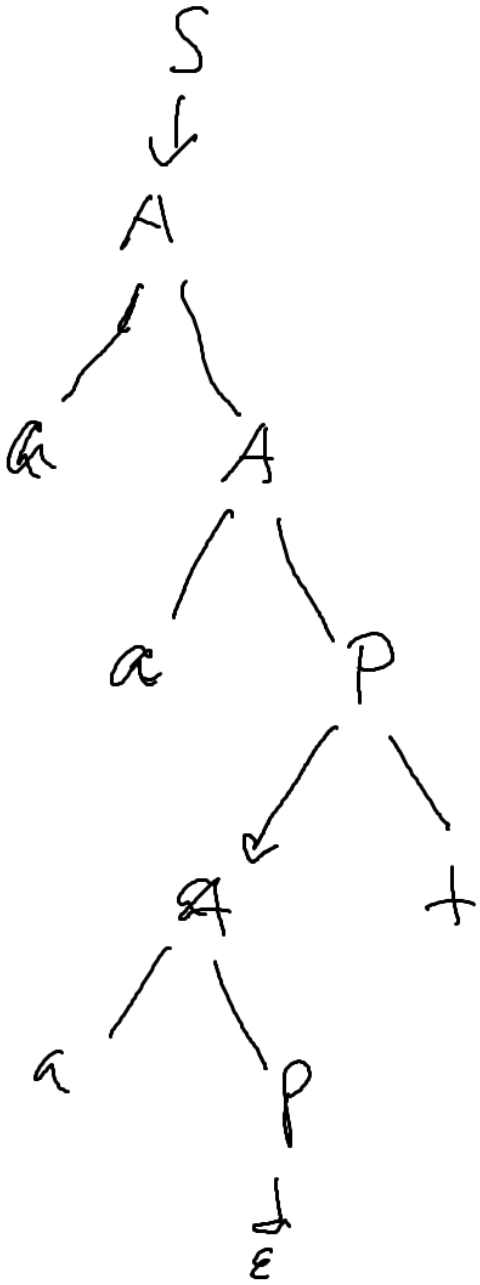
aaa+

b) Det kan ikke lade sig gøre, da der mindst skal være et a mere end +'er

c) ja, $a^+ \ (a +)^*$ er den tilsvarende regex

d)





e)

$$S \rightarrow A$$

$$A \rightarrow a\ A$$

$$A \rightarrow A'$$

$$A' \rightarrow\ a\ P$$

$$P \rightarrow A' +$$

$$P \rightarrow \epsilon$$

2.2

a)

$$Nullable(S) = Nullable(F\ \$)$$
$$= Nullable(F) \wedge Nullable(\$)$$
$$= Nullable(F) \wedge false = false$$

$$Nullable(F) = Nullable(f\ [\ H\ T\ ]$$
$$= Nullable(F) = Nullable(f) \wedge Nullable([\quad) \wedge Nullable(H) \wedge Nullable(T) \wedge Nullable(\quad])$$
$$= Nullable(F) = false \wedge Nullable([\quad) \wedge Nullable(H) \wedge Nullable(T) \wedge Nullable(\quad])$$
$$= false$$

$$Nullable(H) = Nullable(f) \vee Nullable(g)$$
$$= false \vee false = false$$

$$Nullable(T) = Nullable(+H\ T) \vee Nullable(\epsilon)$$
$$= Nullable(+HT) \vee true$$
$$= true$$

b)

$$First(F) = \{f[\ \}$$
$$First(H) = \{f, g\}$$
$$First(T) = \{+\}$$
$$First(S) = First(F) = \{f\}$$

c)

Production	Constraint
$S \rightarrow F\$$	$\{\$\} \subseteq Follow(F)$
$F \rightarrow f\ [\ H\ T\ ]$	$First(T) \subseteq Follow(H)$, $\{]\} \subseteq Follow(T)$
$H \rightarrow f$	

$H \rightarrow g$	
$T \rightarrow + H\ T$	$First(T) \subseteq Follow(H)$
$T \rightarrow \epsilon$	

d)

$$Follow(F) = \{\$\}$$
$$Follow(H) = \{+\}$$
$$Follow(T) = \{]\}$$

e)

$$la(S \rightarrow F\$) = First(F) = \{f\}$$
$$la(F \rightarrow f\ [\ H\ T\ ]) = \{f[\}$$
$$la(H \rightarrow f) = \{f\}$$
$$la(H \rightarrow g) = \{g\}$$
$$la(T \rightarrow +\ H\ T) = \{+\}$$
$$la(T \rightarrow \epsilon) = First(\epsilon) \cup Follow(T) = \{]\}$$

f)

```
function parseS () =
        if input = 'f' or input ='$' then
                parseF() ; match('$')
        else reportError()

function parseF () =
        if input = 'f' then
                match ('f') ;  match ('[') ; parseH() ; parseT() ; match(']')
        else reportError()

function parseH () =
        if input = 'f' then
                match ('f')
        else if input ='g' then
```

```
                    match('g')
            else reportError()

function parseT () =
            if input = ']' or input ='$' then
                    (* do nothing, just return *)
            else if input = '+' then
                    match('+') ; parseH() ; parseT()
            else reportError()
```

Opgave 3.

3.1

$Eval_{Exp}(Exp, vtable, ftable) = case\ Exp\ of$

$\max(Exps)$	$v = Eval_{max}(Exps, vtable, ftable)$ $getvalue(v)$

$Eval_{max}(Exps, vtable, ftable) = case\ Exps\ of$

Exp	$v_1 = Eval_{Exp}(Exp, vtable, ftable)$ $if\ v_1\ is\ an\ interger\ then$ $\quad v_1$ $else\ error()$
$Exp, Exps$	$v_1 = Eval_{Exp}(Exp, vtable, ftable)$ $if\ v_1 = INT\_MAX\ \ then$ $\quad v_1$ $v_2 = Eval_{max}(Exps, vtable, ftable)$ $if\ v_1\ and\ v_2\ are\ integers\ then$ $\quad if\ v_1 < v_2\ then$ $\quad\quad v_2$ $\quad else\ v_1$

	$else\ error()$

3.2

$Check_{Exp}(Exp, vtable, ftable) = case\ Exp\ of$

$acc(f, Exp_{arr}, Exp_b)$	$t_{arr} = Check_{Exp}(Exp_{arr}, vtable, ftable)$ $t_{el} = case\ t_{arr}\ of$ $\quad \mid Array(t_1) \rightarrow t_1$ $\quad \mid Otherwise \rightarrow error()$ $t_b = Check_{Exp}(Exp_b, vtable, ftable)$ $t_f = lookup(ftable, name(f))$ $case\ t_f\ of$ $\quad \mid unbound \rightarrow error()$ $\quad \mid ([t_{in,a}, t_{in,b}] \rightarrow t_{out}) \rightarrow$ $\quad\quad if\ t_{in,a} == t_{el}\ and\ t_{in,b} == t_b\ and\ t_{out} == t_b\ then$ $\quad\quad\quad t_{out}$ $\quad\quad else\ error()$ $\quad \mid otherwise \rightarrow error()$

Opgave 4.

4.1

Går ud fra j ikke er med, da den ikke bliver brugt i koden, selvom der står j mapper til v_1

$$t_1 = v_2$$
$$t_2 = t_1 + 4$$
$$v_0 = M[t_2]$$

$Label\ startWhile$:

$$t_3 = v_0$$
$$t_4 = 0$$

$if\ t_3 > t_4\ then\ \ trueLabel\ \ else\ \ endWhile$

$Label\ trueLabel$:

$t_5 = v_2$

$t_6 = v_0$

$t_7 = t_5 + t_6$

$t_8 = M[t_7]$

$t_9 = 0$

$if\ t_8\ != t_9\ then\ \ bodywhile\ \ else\ endWhile$

$Label\ bodyWhile$:

$t_{10} = v_0$

$t_{11} = 1$

$t_{12} = CALL\_f_0(t_{10}, t_{11})$

$v_0 = t_{12}$

$Label\ endWhile$:

$t_{13} = v_0$

$t_{14} = v_2$

$t_{15} = t_{13} + t_{14}$

$M[t_{15}] = t_{13}$

4.2

$slt\quad R1, r_a, r_c$

$beq\quad R1, R0, lab_2$

$Label\ lab_1$

$addi\ \ r_a, r_a, 471$

$add\ \ r_b, r_a, r_b$

$lw\quad r_x, 0(r_a)$

$lw\quad r_y, 0(r_b)$

4.3

```
Let t1 = newReg ”expt_L”
Let t2 = newReg ”expt_R”
Let 1Reg = newReg ”1Reg”
Let negativReg = newReg ”negativReg”

Let code1 = compileExp e1 vtable t1
Let code2 = compileExp e2 vtable t2

Let newLab = ”negativExpt”
Let newLab = ”0Expt”
Let newLab = ”body”
Let newLab = ”endLab”

Code2 @
[ Mips.SLT (negativReg, t2, RZ)
; Mips.BEQ (negativReg, RZ, negativExpt)
; Mips.BEQ (t2, RZ, 0Expt)
; Mips.LI (1Reg, 1)
; Mips.LABEL body
; Mips.BEQ (t2, RZ, endLab)
; Mips.MUL (place, t1, t1)
; Mips.SUB (t2, t2, 1Reg)
; Mips.J body
; Mips.Label negativExpt
; Mips.LI (RN5, pos)
; Mips.LA(RN6. ” _Msg_IllegalExpt_”)
; Mips.J ”_RuntimeError_”
; Mips.LABEL 0Expt
; Mips.ADDI (place, t1, 0)
```

; Mips.LABEL endLab

Opgave 5

5.1

a) c, da d bliver dræbt og b bliver læst

b) a, da der ikke er noget, som bliver dræbt, og e bliver læst

og d, da c bliver dræbt, og e og c bliver læst

5.2

a)

i	$succ[i]$	$gen[i]$	$kill[i]$
1	2		
2	3, 9	Y	
3	4		
4	5	Y, x	S
5	6	Y	T
6	7	S, x	X
7	8	T, y	Y
8	1		
9	10		
10		X	

b)

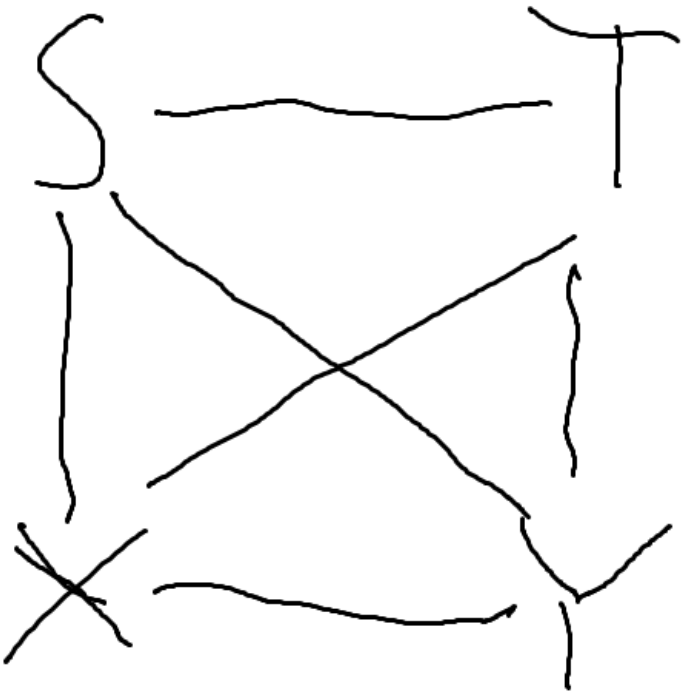
	Iteration 1		Iteration 2	
i	$out[i]$	$in[i]$	$out[i]$	$in[i]$
1	Y, x	Y, x	Y, x	Y, x

2	Y, x	Y, x	Y, x	Y, x
3	Y, x	Y, x	Y, x	Y, x
4	Y, s, x	Y, x	Y, s, x	Y, x
5	T, y, s, x	Y, s, x	T, y, s, x	Y ,s, x
6	T, y	T, y, s, x	T, y, x	T, y, s, x
7		T, y	Y, x	T, y x
8			Y, x	Y, x
9	X	X	x	X
10		X		X

c)

i	$out[i]$	$kill[i]$	Interferes with
4	Y, x	S	Y, x
5	Y ,s, x	T	Y, s, x
6	T, y, s, x	X	T, y ,s
7	T, y x	Y	T, x

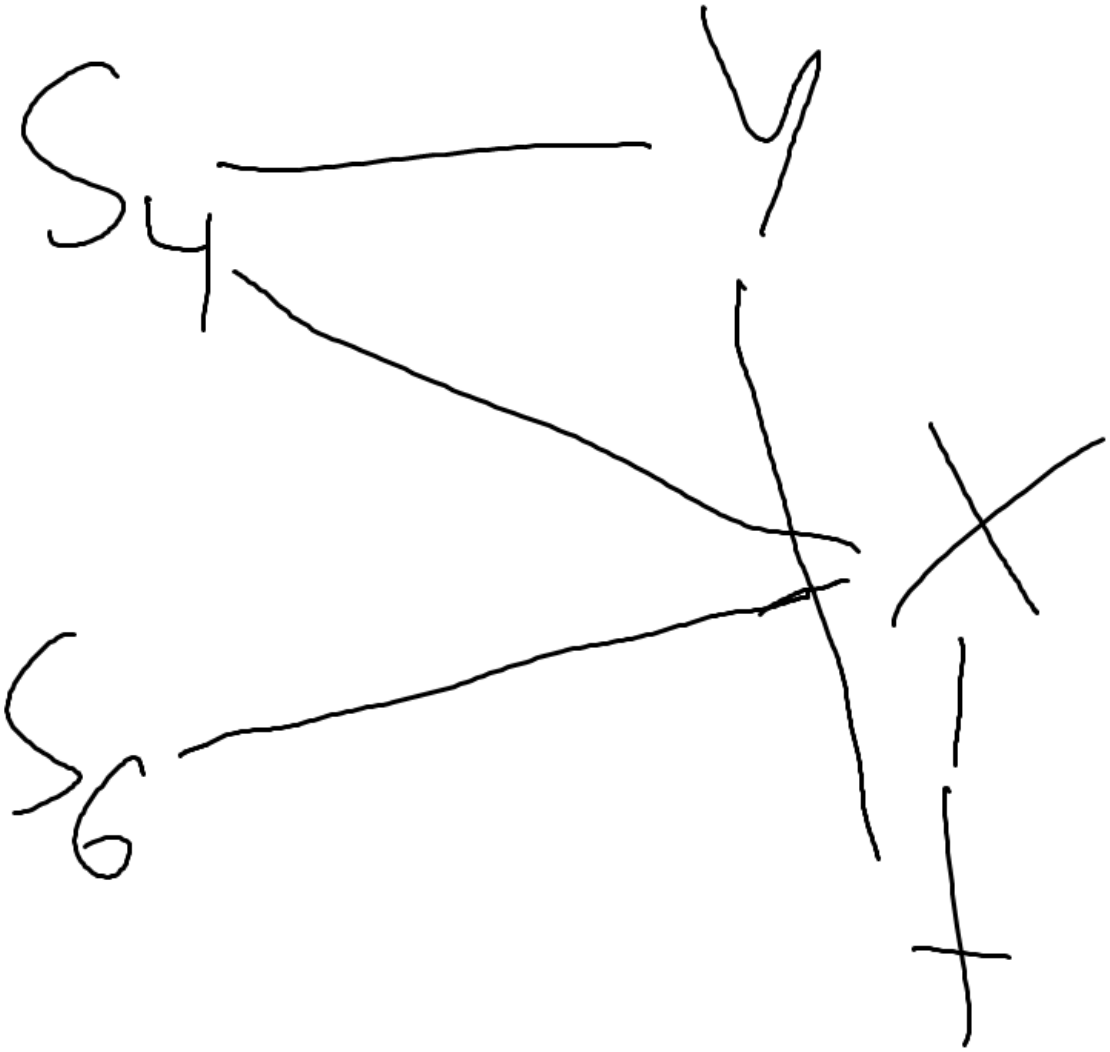
d)



e)

Node	Neighbors	color
T		1
X	T	2
Y	T, x	3
S	T, x, y	Spill

Spill:



Node	Neighbors	Color
S4		1
T		1
Y	T	2
X	T,y	3
S6	X	1