1) a) No, this specification does not define a function, since it does not specify what the value of $f(0)$ should be, i.e. there is not a *unique* function satisfying the specification. For instance, both $\{(0,0),(1,1),(2,2),(3,3),\ldots\}$, that is the function $f(x) = x$, and $\{(0,1),(1,2),(2,3),(3,4),\ldots\}$, that is the function $f(x) = x+1$, satisfy the specification.

b) Yes, the specification determines a unique value for every natural number. The first conditions yields $(0,17)$. Using this the second condition yields $(1,17+1) = (1,18)$. Using it again yields $(2,18+1) = (2,19)$. Continuing like this we obtain unique values for every natural number in the domain: $f = \{(0,17),(1,18),(2,19),(3,20),\ldots\}$, that is the function $f(x) = x+17$.

c) No, this specification does not define a function, since repeatedly using the equations we calculate that $f(18)+18 = f(17+1)+1+17 = f(17)+17 = f(16+1)+1+16 = f(16)+16 = \ldots = f(0)+0 = 17$, but there is no natural number such that when we add 18 to it, we obtain 17, i.e., there does not *exist* a function satisfying the specification.

d) Yes, this specification defines the function $\{(0,1),(1,1),(2,2),(3,3),(4,5),(5,8),(6,13),(7,21),\ldots\}$ where a value is the sum of the previous two; the Fibonacci numbers $f(i) = F_i$.

e∗) Yes, this specification defines the function $\{(0,1),(1,3),(2,7),(3,61),(4,2^{2^{2^{2^{2^{2^{2}}}}}}-3),(5,2{\uparrow}{\uparrow}{\uparrow}(n+3)-3),\ldots\}$. Here we used Knuth's uparrow notation to express the extremely large value for 5. There is no hope to compute this number and write it in digits, even using as many resources as needed, as there are not enough atoms in the universe. That still the specification defines a function can be seen by noting that to compute the value of $g(m,n)$ we only rely on *earlier* computed values for g, namely of values $g(m',n')$ where either $m' < m$ or ($m' = m$ and $n' < n$); g is known as the Ackermann function.

f) Yes, this specification defines a function since $f(n) = 1+\max(f_b(n), f_d(n))$ where f_b and f_d are the functions defined in items b) and d) respectively. (In case also exercise e∗) is included, then $f(n) = 1+\max(f_b(n), f_d(n), f_e(n))$.)

g) No, this specification does not define a function, since there is more than one function meeting it (it is not *unique*). For instance, both the constant 0 function $f(n) = 0$, that is $\{(0,0),(1,0),(2,0),\ldots\}$ and the constant 1 function $f(n) = 1$, that is $\{(0,1),(1,1),(2,1),\ldots\}$ satisfy the specification since $0 = 0^2$ and $1 = 1^2$.

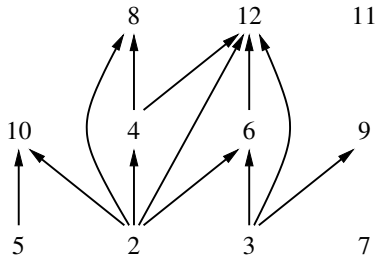
2) We compute for B:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

From the resulting matrix we read off that $a\ R^+\ c$ and $d\ R^+\ b$ hold, but $c\ R^+\ a$ does not. The matrices in the computation above can be obtained from the computation from B in exercise 3) of the 1st sheet as follows: map all elements with value ∞ and those on the diagonal to

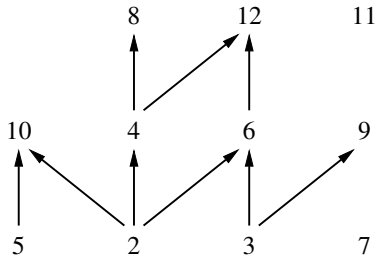
0, and all other elements to 1. (The special treatment of the diagonal would be unnecessary in case we would be interested in computing the *reflexive*–transitive closure R^* of R; see the slides. The special treatment of the diagonal is possible thanks to R^+ being irreflexive here.)

3) • We may draw G_T as



To see that $|_T$ is irreflexive observe that G_T has no loops, and to show that it is transitive we need to show that for every path from n to m of length 2 in G_T, there is an edge from n to m. There are only four paths of length 2 in G_T, namely $2 \to 4 \to 8$, $2 \to 4 \to 12$, $2 \to 6 \to 12$, $3 \to 6 \to 12$, and since we indeed have the required edges $2 \to 8$, $2 \to 12$, $2 \to 12$, $3 \to 12$. Alternatively, we can reason algebraically: if $x \cdot x' = y$ and $y \cdot y' = z$, then $x \cdot (x' \cdot y') = (x \cdot x') \cdot y' = y \cdot y' = z$, as desired. From this we conclude, since indeed $x' \cdot y' \in T$ ($2 \leq x' \cdot y'$ since $2 \leq x', y'$ and $x' \cdot y' \leq 12$ since $x' \cdot y' \mid_T z \leq 12$).

• After consecutively removing the edges $2 \to 8$, $2 \to 12$, and $3 \to 12$, no edges can be removed anymore, yielding the graph G'_T:



The procedure terminates since in every iteration an edge is removed, and there were only finitely many such. The order of removing edges e and e' because of paths p and p' can only matter if both $e \in p'$ and $e' \in p$, but that is impossible as G_T is acyclic. G'_T is a *transitive reduct* of G_T: $G'^+_T = G_T$ but G'_T is 'nowhere' transitive; cf. *Hasse diagrams*.

4∗) Consider the Haskell code:

```
merge l@(x:l') m@(y:m') = if (x < y) then x:merge l' m else y:merge l m'
merge l m = l++m
mergeSort m = if l == 0 then m else merge (mergeSort (fst s)) (mergeSort (snd s)) where
  l = length m `div` 2
  s = splitAt l m
bubble x [] = (x,[])
bubble x (y:m) = (\p -> ((fst p),(max x y):(snd p))) (bubble (min x y) m)
bubbleSort (x:l) = let (y,m) = bubble x l in y:bubbleSort m
bubbleSort l = l
```

`mergeSort` and `bubbleSort` both implement the sorting function on lists of `Int`s, but their *runtimes* are different. Setting `:set +s` in ghci, my times for sorting `[1000,999..1]` are 0.03 secs respectively 0.35 secs. (Their respective *complexities* are $O(n \log n)$ and $O(n^2)$.)