1. (12 p)

Lösning:

(a) `max3 :: Int -> Int -> Int -> Int`
(eller samma lösning med `Int` utbytt mot `Integer`)

(b)
```
max3 8 (3+4) 9
==> 8 >= (3+4) && 8 >= 9  Rad 1 utvärderas.
==> 8 >= 7 && 8 >= 9      Rad 1 utvärderas.
==> True && False         Rad 1 utvärderas.
==> False                 Rad 1 utvärderas.
==> 7 >= 9                Rad 2 utvärderas.
==> False                 Rad 2 utvärderas.
==> True                  Rad 3 utvärderas.
==> 9                     Rad 3 utvärderas.
9
```

(c)
1. Värden evalueras först när det behövs s.k. Call-by-name
   Exempel: `(3+4)` i det andra beräkningssteget
2. Sharing används och konsekvensen är att värden beräknas högst en gång
   Exempel: `(3+4)` i beräkningssteg 5 är redan beräknat i steg 2.

(d) En anonym funktion som tar ett heltal som indata och returnerar det heltal som är det största av 1, 2 och indata.

Rättning:

På varje uppgift:
0p är minsta poäng som kan ges på varje deluppgift.
-2p logiska fel
-1p för varje slarvfel (men högst 1p för samma typ av slarvfel)
Inga avdrag för slarvfel där innebörden tydligt framgår t.ex. Intger istället Integer

Uppgiftsspecifik bedömning:

(a) -1p för varje Int/Integer som saknas
-2p om typen är Num

(b)

(c) 1p för varje rätt delsvar:
- Säger något om att "värden evalueras först när det behövs" eller "call-by-name används" eller "Outermost reduction"
- Korrekt exemplifiering av call-by-name
- Säger något om att "Sharing används" eller "värden beräknas högst en gång" eller "pekare till värdet används"
- Korrekt exemplifiering av sharing

(d) -2p nämner inte att en funktion returneras
-2p uppger felaktig indata till den anonyma funktionen
-2p uppger felaktigt resultat vid anrop av den anonyma funktionen

2. (8 p)

Lösning:

(a) Rätt alternativ är `e1`.

(b) Möjlig lösning:

```
nChars :: Answer -> Int -> Int
nChars [] count = count                                    -- basfall
nChars (_:charList) count = nChars charList (count + 1)    -- rekursivt anrop
```

(c) Möjlig lösning:

```
totnChars :: Entryt -> Int
totnChars entry = myLoop (answers entry)  -- beräkningen startas med rätt indata

myLoop :: [Answer] -> Int
myLoop [] = 0                                              -- basfall
myLoop (x:rest) = (nChars x 0) + (myLoop rest)             -- rekursivt anrop
```

Rättning:

På varje uppgift:
0p är minsta poäng som kan ges på varje deluppgift.
-2p logiska fel
-1p för varje slarvfel (men högst 1p för samma typ av slarvfel)
Inga avdrag för slarvfel där innebörden tydligt framgår t.ex. Intger istället Integer
Uppgiftsspecifik bedömning:

(a) Rätt alternativ ger 1p övriga svar noll poäng.

(b) Typsignatur behöver ej anges.
-2p basfall saknas
-2p rekursivt anrop saknas
-2p ackumulator-variabeln returneras inte i basfallet
-2p felaktig uppdatering av argument i det rekursiva anropet

(c) Typsignatur behöver ej anges.
-1p om bara hjälpfunktionen skrivits, ej den yttre som tar ett `Entryt`
-2p felaktig indata till hjälpfunktionen
-2p felaktig indata till `nChars`
-2p basfall saknas
-2p rekursivt anrop saknas
-2p 0 returneras inte i basfallet
-2p felaktig uppdatering av argument i det rekursiva anropet