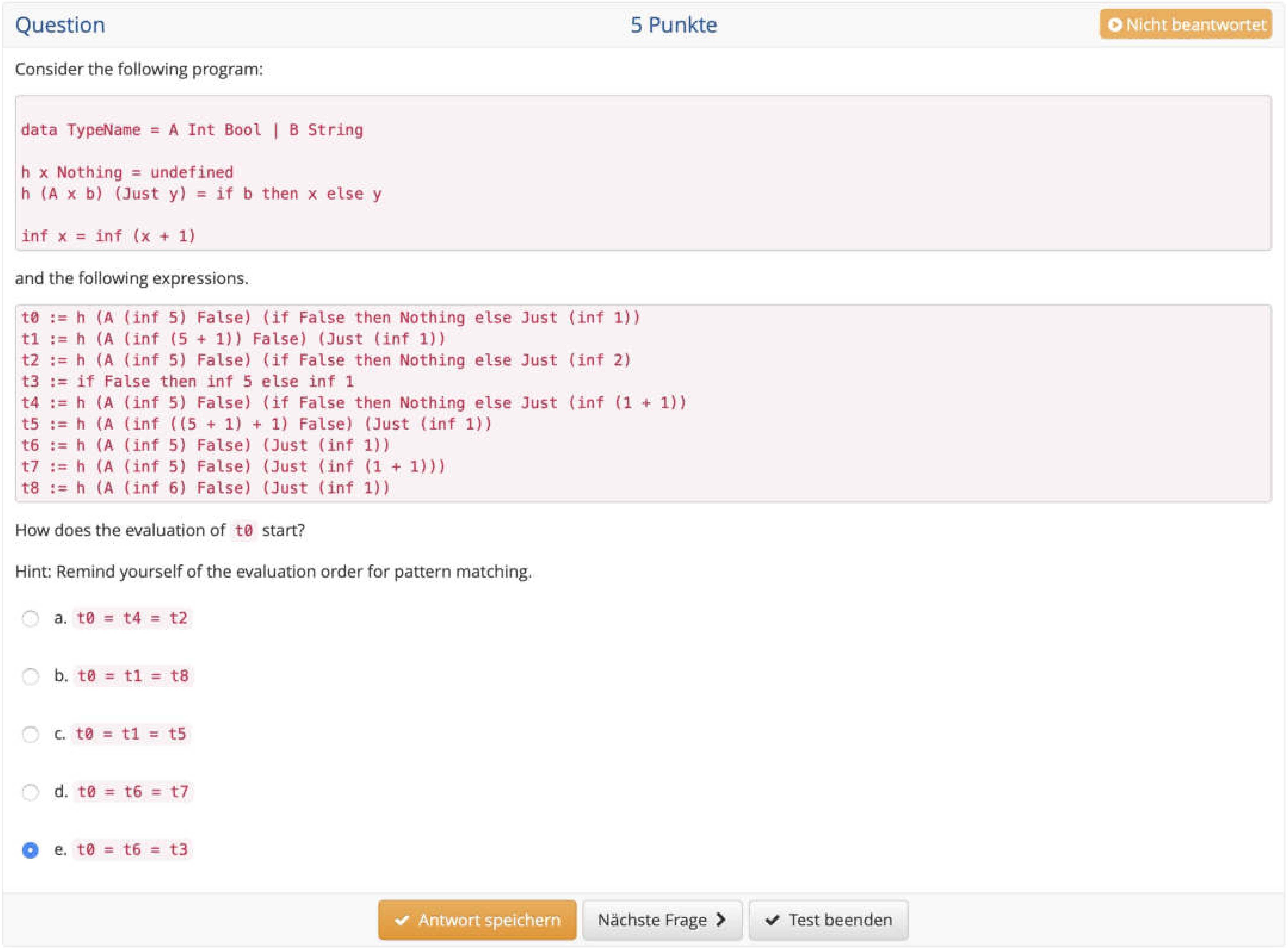


Question

5 Punkte

Consider the following program:

```
data TypeName = A Int Bool | B String

h x Nothing = undefined
h (A x b) (Just y) = if b then x else y

inf x = inf (x + 1)
```

and the following expressions.

```
t0 := h (A (inf 5) False) (if False then Nothing else Just (inf 1))
t1 := h (A (inf (5 + 1)) False) (Just (inf 1))
t2 := h (A (inf 5) False) (if False then Nothing else Just (inf 2)
t3 := if False then inf 5 else inf 1
t4 := h (A (inf 5) False) (if False then Nothing else Just (inf (1 + 1))
t5 := h (A (inf ((5 + 1) + 1) False) (Just (inf 1))
t6 := h (A (inf 5) False) (Just (inf 1))
t7 := h (A (inf 5) False) (Just (inf (1 + 1)))
t8 := h (A (inf 6) False) (Just (inf 1))
```

How does the evaluation of `t0` start?

Hint: Remind yourself of the evaluation order for pattern matching.

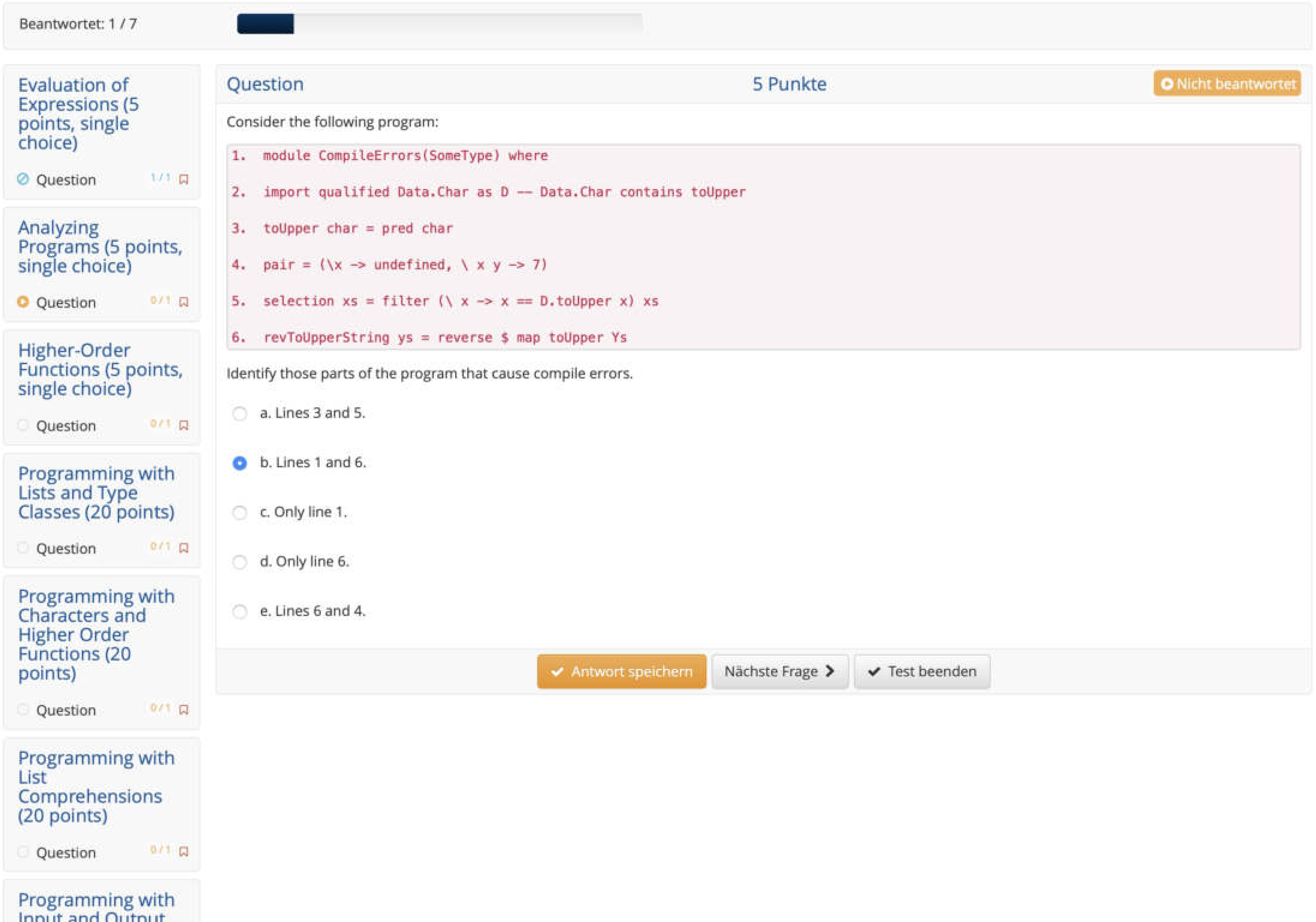
a. `t0 = t4 = t2`

b. `t0 = t1 = t8`

c. `t0 = t1 = t5`

d. `t0 = t6 = t7`

[✓ e. `t0 = t6 = t3`]


Beantwortet: 1 / 7

Evaluation of Expressions (5 points, single choice)

Question 1 / 1

Analyzing Programs (5 points, single choice)

Question 0 / 1

Higher-Order Functions (5 points, single choice)

Question 0 / 1

Programming with Lists and Type Classes (20 points)

Question 0 / 1

Programming with Characters and Higher Order Functions (20 points)

Question 0 / 1

Programming with List Comprehensions (20 points)

Question 0 / 1

Programming with Input and Output

Question

5 Punkte

Nicht beantwortet

Consider the following program:

```
1.  module CompileErrors(SomeType) where

2.  import qualified Data.Char as D -- Data.Char contains toUpper

3.  toUpper char = pred char

4.  pair = (\x -> undefined, \ x y -> 7)

5.  selection xs = filter (\ x -> x == D.toUpper x) xs

6.  revToUpperString ys = reverse $ map toUpper Ys
```

Identify those parts of the program that cause compile errors.

a. Lines 3 and 5.

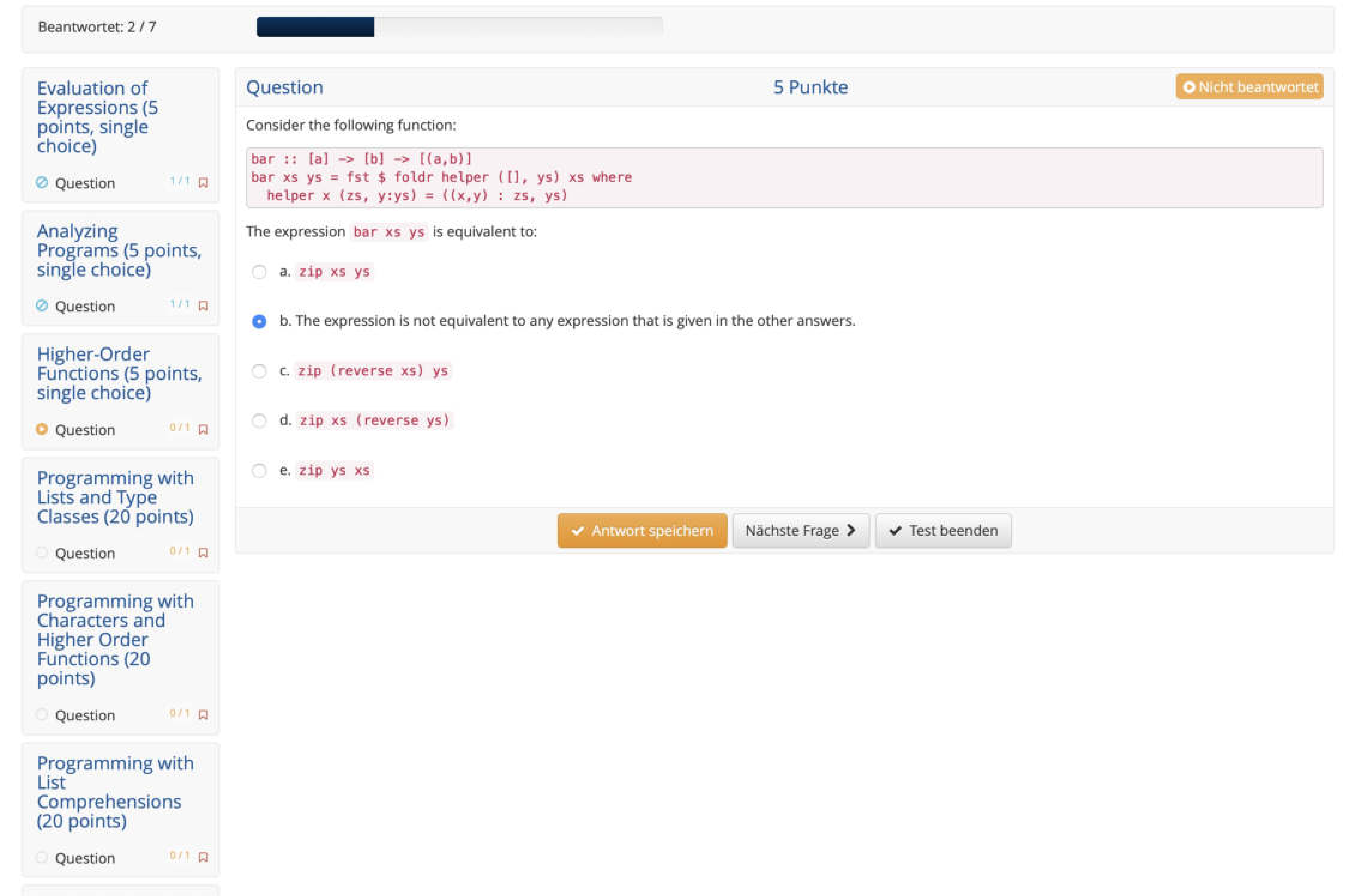
[✓ b. Lines 1 and 6.]

c. Only line 1.

d. Only line 6.

e. Lines 6 and 4.

Antwort speichern | Nächste Frage | Test beenden



Beantwortet: 2 / 7

Evaluation of Expressions (5 points, single choice)

Question 1 / 1

Analyzing Programs (5 points, single choice)

Question 1 / 1

Higher-Order Functions (5 points, single choice)

Question 0 / 1

Programming with Lists and Type Classes (20 points)

Question 0 / 1

Programming with Characters and Higher Order Functions (20 points)

Question 0 / 1

Programming with List Comprehensions (20 points)

Question 0 / 1

Question

5 Punkte

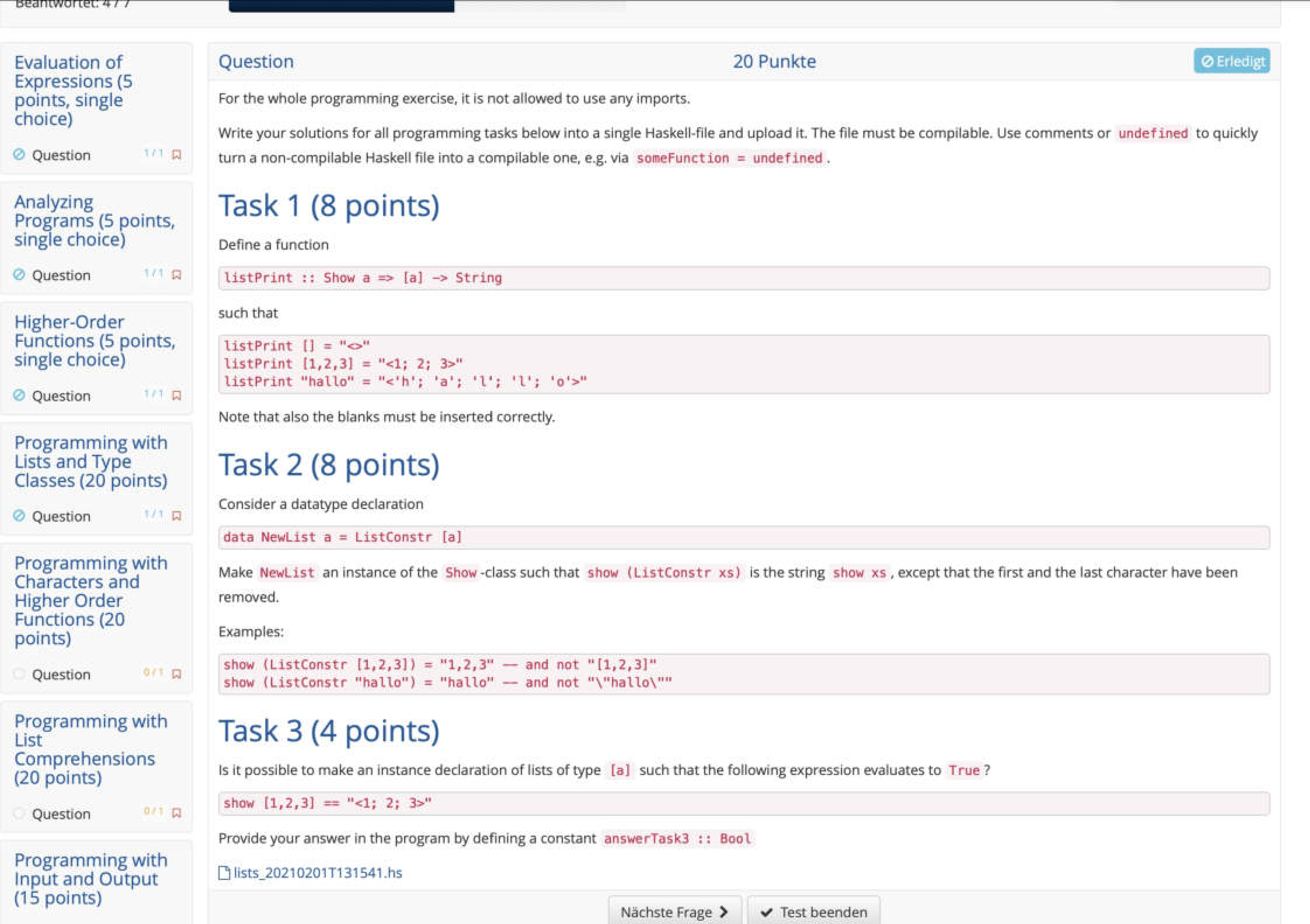
Nicht beantwortet

Consider the following function:

```haskell
bar :: [a] -> [b] -> [(a,b)]
bar xs ys = fst $ foldr helper ([], ys) xs where
  helper x (zs, y:ys) = ((x,y) : zs, ys)
```

The expression `bar xs ys` is equivalent to:

- a. `zip xs ys`
- [✓ b. The expression is not equivalent to any expression that is given in the other answers.]
- c. `zip (reverse xs) ys`
- d. `zip xs (reverse ys)`
- e. `zip ys xs`



Beantwortet: 4 / 7

Question 20 Punkte

For the whole programming exercise, it is not allowed to use any imports.

Write your solutions for all programming tasks below into a single Haskell-file and upload it. The file must be compilable. Use comments or `undefined` to quickly turn a non-compilable Haskell file into a compilable one, e.g. via `someFunction = undefined`.

Task 1 (8 points)

Define a function

```
listPrint :: Show a => [a] -> String
```

such that

```
listPrint [] = "<>"
listPrint [1,2,3] = "<1; 2; 3>"
listPrint "hallo" = "<'h'; 'a'; 'l'; 'l'; 'o'>"
```

Note that also the blanks must be inserted correctly.

Task 2 (8 points)

Consider a datatype declaration

```
data NewList a = ListConstr [a]
```

Make `NewList` an instance of the `Show`-class such that `show (ListConstr xs)` is the string `show xs`, except that the first and the last character have been removed.

Examples:

```
show (ListConstr [1,2,3]) = "1,2,3" -- and not "[1,2,3]"
show (ListConstr "hallo") = "hallo" -- and not "\"hallo\""
```

Task 3 (4 points)

Is it possible to make an instance declaration of lists of type `[a]` such that the following expression evaluates to `True`?

```
show [1,2,3] == "<1; 2; 3>"
```

Provide your answer in the program by defining a constant `answerTask3 :: Bool`

lists_20210201T131541.hs

Question 20 Punkte Erledigt

For the whole exercise, it is not allowed to use any imports.

Write your solutions for all programming tasks below into a single Haskell-file and upload it. The file must be compilable. Use comments or `undefined` to quickly turn a non-compilable Haskell file into a compilable one, e.g. via `someFunction = undefined`.

Task 1 (7 points)

Define a function `minValue :: Ord b => (a -> b) -> [a] -> a` such that `minValue f xs` is any element `x` of `xs` such that `f x <= f y` for all `y` in `xs`. Here, you can assume that `xs` is non-empty.

For example: `minValue negate [3,7,5] = 7`.

Task 2 (9 points)

Consider the following type definition for representing items in a store

```
type Item a = (a, Integer)   -- (item identifier, weight)
```

where in a list of items, each item identifier is unique and all weights are positive.

Define a function

```
triples :: Ord a => [Item a] -> [((a, a, a), Integer)]
```

such that the resulting list consists of precisely those entries `((i1, i2, i3), w)` that satisfy the following criteria:

- `i1`, `i2`, `i3` are item identifiers that occur in the input list,
- the item identifiers are sorted: `i1 > i2 > i3`,
- the number `w` is the total weight of the three items, and
- `w` is at least 143.

Example:

```
triples [
triples :: Ord a => [Item a] -> [((a, a, a), Integer)]
```

such that the resulting list consists of precisely those entries `((i1, i2, i3), w)` that satisfy the following criteria:

- `i1`, `i2`, `i3` are item identifiers that occur in the input list,
- the item identifiers are sorted: `i1 > i2 > i3`,
- the number `w` is the total weight of the three items, and
- `w` is at least 143.

Example:

```
triples [
    ("Oranges", 30),
    ("Bananas", 58),
    ("Iron", 20),
    ("Paper", 64)
  ]
  = [(("Paper", "Oranges", "Bananas"), 152)]
```

Task 3 (4 points)

Combine `triples` and `minValue` to write a function

```
bestCombination :: Ord a => [Item a] -> (a, a, a)
```

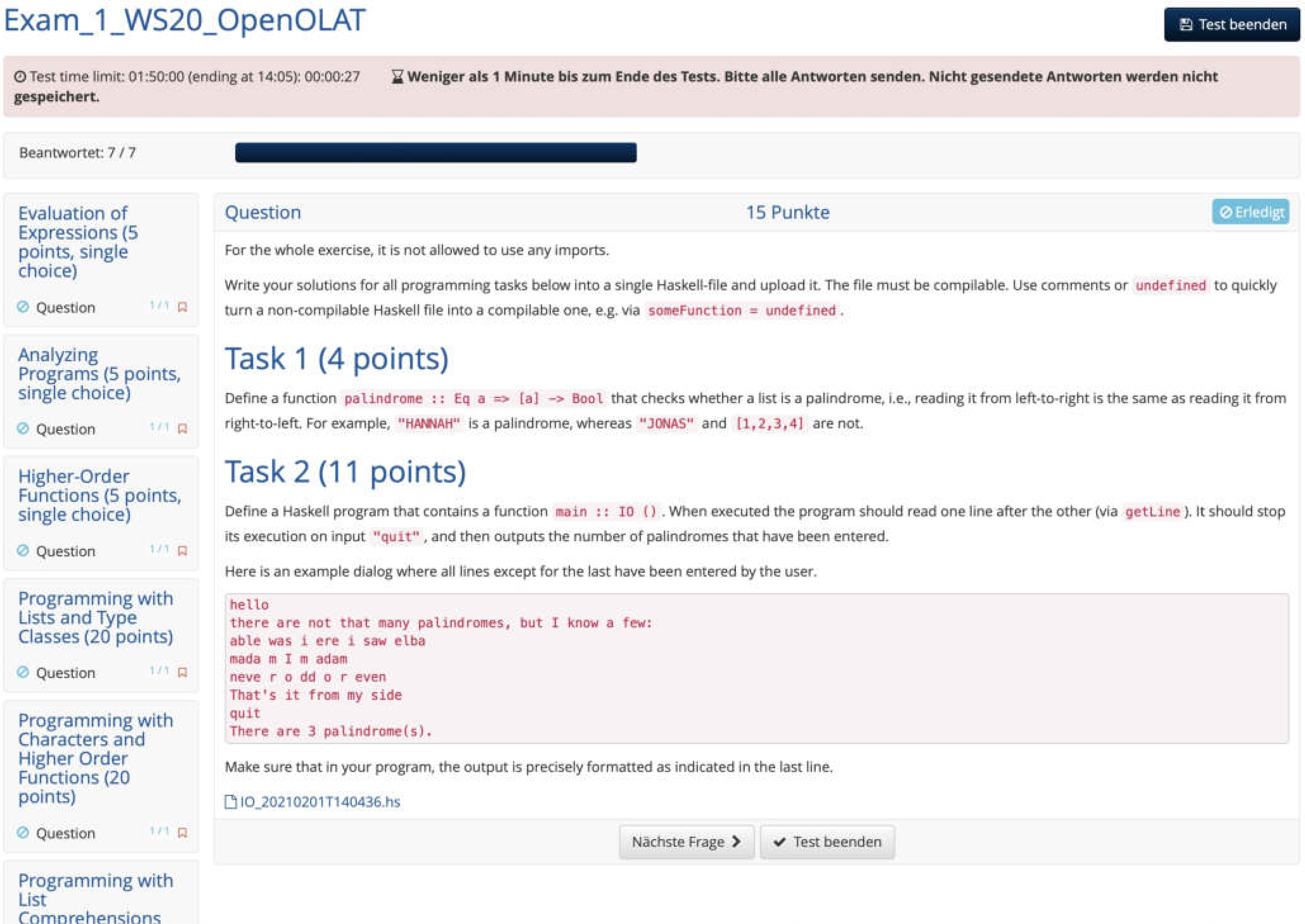
that computes some combination of exactly three items such that the total weight is at least 143 and such that the total weight is minimal among all such combinations. You can assume that the input list contains at least three different items whose total weight is at least 143.

Example:

```
bestCombination [
    ("Ding", 49),
    ("Foo", 44),
    ("Bar", 58),
    ("Word", 64)
  ]
  = ("Foo", "Ding", "Bar")
```

comprehensions_20210201T135017.hs

Nächste Frage ›   ✔ Test beenden



Exam_1_WS20_OpenOLAT

Test beenden

Test time limit: 01:50:00 (ending at 14:05): 00:00:27 **Weniger als 1 Minute bis zum Ende des Tests. Bitte alle Antworten senden. Nicht gesendete Antworten werden nicht gespeichert.**

Beantwortet: 7 / 7

Evaluation of Expressions (5 points, single choice)

Question 1/1

Analyzing Programs (5 points, single choice)

Question 1/1

Higher-Order Functions (5 points, single choice)

Question 1/1

Programming with Lists and Type Classes (20 points)

Question 1/1

Programming with Characters and Higher Order Functions (20 points)

Question 1/1

Programming with List Comprehensions

Question 15 Punkte Erledigt

For the whole exercise, it is not allowed to use any imports.

Write your solutions for all programming tasks below into a single Haskell-file and upload it. The file must be compilable. Use comments or `undefined` to quickly turn a non-compilable Haskell file into a compilable one, e.g. via `someFunction = undefined`.

Task 1 (4 points)

Define a function `palindrome :: Eq a => [a] -> Bool` that checks whether a list is a palindrome, i.e., reading it from left-to-right is the same as reading it from right-to-left. For example, `"HANNAH"` is a palindrome, whereas `"JONAS"` and `[1,2,3,4]` are not.

Task 2 (11 points)

Define a Haskell program that contains a function `main :: IO ()`. When executed the program should read one line after the other (via `getLine`). It should stop its execution on input `"quit"`, and then outputs the number of palindromes that have been entered.

Here is an example dialog where all lines except for the last have been entered by the user.

```
hello
there are not that many palindromes, but I know a few:
able was i ere i saw elba
mada m I m adam
neve r o dd o r even
That's it from my side
quit
There are 3 palindrome(s).
```

Make sure that in your program, the output is precisely formatted as indicated in the last line.

IO_20210201T140436.hs

Nächste Frage

Test beenden