CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
TDA555	Introduction to functional programming			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
TDA555-0008-KGS		03/01-2023	9	5

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	U	
2	X	G	
3	X	G	
4	X	G	
5	X	G	
6	X	G	
7	X	G-	
8	X	G	
9	X	G	
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng på tentamen		6/2	

CHALMERS	Anonymous code	Points for question (to be filled in by teacher)	Consecutive page no. Löpande sid nr 1
	Anonym kod TDA555-0008-K6S	Poäng på uppgiften (ifylles av lärare) U	Question no. Uppgift nr 1

1)

a)

f 2 "A"

= map g 2 "A"

"A" : map g 1 "A"

"A" : "A" : map g 0 "A"

"A" : "A" : []

["A", "A"]

b) f :: Num a => a -> [b] -> [[b]]

g :: Num a => a -> b -> [b]

Int

CHALMERS | Anonymous code / Anonym kod: TDA555-0008-KGS | Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare): 6 | Consecutive page no. / Löpande sid nr: 2 | Question no. / Uppgift nr: 2

2

a)
```
pell :: Int -> Integer
pell 0 = 0
pell 1 = 1
pell n = 2 * (pell (n-1)) + pell (n-2)
```

b)
```
approx :: Int -> Double
approx n = (fromIntegral (pell (n-1)) + d) / d
    where d = fromIntegral (pell n)
```

CHALMERS	Anonymous code	Points for question (to be filled in by teacher)	Consecutive page no. Löpande sid nr 3
	Anonym kod TDA555-0008-KG6	Poäng på uppgiften (ifylles av lärare) 4	Question no. Uppgift nr 3

3

```
mapMindMap :: (String -> String) -> MindMap -> MindMap
mapMindMap f mm = case mm of
    Topic s -> Topic (f s)
    Branch s xs -> Branch f s  map (mapMindMap f) xs
```

CHALMERS | Anonymous code | Points for question (to be filled in by teacher) | Consecutive page no. Löpande sid nr 4

Anonym kod: TDA555-0008-KGS | Poäng på uppgiften (ifylles av lärare): 6 | Question no. Uppgift nr 4

4

```
readPerson :: IO Person
readPerson = do
putStrLn "Name :"
pName <- getLine
putStrLn "Age :"
pAge <- readLine
putStrLn "Partner (y/n):"
pAnswer <- getLine
if pAnswer == "y"
        then  putStrLn "Partner Name:"
              ppName <- getLine
             let person = Person {name = pName, age = pAge
                 , partner = Just ppName}
             return $ show person
  Else
     let person = Person {name = pName, age = pAge, partner = Nothing}
     return $ show person
```

2

5)

a)
```
valid :: BillBoard -> Bool
valid (BB (x,y) xs) = checkBoolList [x>=i && y>=j | (i,j) <- xs]

checkBoolList :: [Bool] -> Bool
checkBoolList [] = True
checkBoolList (x:xs)
    | x == False = False
    | otherwise = checkBoolList xs
```
and! ≥0 ?

b)
```
move :: Int -> BillBoard -> BillBoard
move 0 (BB (x,y) xs) = (BB (x,y) xs)
move n (BB (x,y) xs) = (BB (x,y) [(i+n, j) | (i,j) <- xs, x>=i+n])
```

CHALMERS	Anonymous code	Points for question (to be filled in by teacher)	Consecutive page no. Löpande sid nr 6
	Anonym kod TDA555-0008-KGS	Poäng på uppgiften (ifylles av lärare) 6	Question no. Uppgift nr 6

6

a)

```
prop_larger :: Int -> Bool
prop_larger n
  | n < 0 = True
  | n > 50 = True
  | otherwise = pell n > pell (n-1)
```

b)

```
prop_identity :: Int -> Bool
prop_identity n
  | n < 0 = True
  | n > 50 = True
  | otherwise = (pell (n+1) * pell (n-1) - (pell n) ^ 2) == (-1) ^ n
```

CHALMERS | Anonymous code | Points for question (to be filled in by teacher) | Consecutive page no. Löpande sid nr 7

Anonym kod: TDA555-0008-KGS | Poäng på uppgiften (ifylles av lärare) G- | Question no. Uppgift nr 7

7

not either, but both

```
data Game = Player1 Player Board | Player2 Player Board
data Player = P { name :: String,
                  points :: Int }
data Board = B { size :: (10,10), not a type
                 cells :: [cell] ,
                 ships :: [Ship] }
data Cell = C { hasShip :: Bool,
                hasBeenHit :: Bool,
                coord :: Coord }
data Ships = Carrier | BattleShip | Destroyer | Submarine | Patrol
type Coord = (Int, Int)
```

2

8)

```
run :: [[Key]] -> String
run keys = exec keys [] [] []

exec :: [Key] -> String -> String -> String -> String
exec [] list1 list2 string = list1 ++ list2
exec (x:xs) list1 list2 string = case x of
        Chr c -> exec xs (list1 ++ c) list2 string
        Del   -> exec xs reverse(tail(reverse list1)) list2 string
                 -- fails on empty list
        GoLeft -> if list1 == []
                    then exec xs list1 list2 string
                  Else exec xs (drop 1 list1) (take 1 list1) ++ list2 string
        GoRight -> if list2 == []
                     then exec xs list1 list2 string
                   Else exec xs list1 ++ (take 1 list2) (drop 1 list2) string
        Copy n -> exec xs list1 list2
                    reverse (take n (reverse list1))

        Paste -> exec xs list1 ++ string (list2) string
```

2

9

```
whitin :: String -> HTML -> [[Tag]]
whitin s xs = [getContent 0 s (drop i xs) | (h,i) <- zip xs [0..], h == Open s]

getContent :: Int -> String -> HTML -> [Tag]
getContent n s (t:ts) = case t of

    Text st -> t : getContent n s ts

    Open st -> if st == s
                 then t : getContent (n+1) s ts
               Else t : getContent n s ts
    Close st -> if st == s
                  then if n-1 == 0
                         then t
                       Else t : getContent (n-1) s ts
                Else t : getContent n s ts
```