

# TENTAMEN I CD5100 FUNCTIONAL PROGRAMMING

Friday Oct. 27th, 2006, 14:30 – 19:30

Course literature is not allowed, and nor is other aids that in some way can replace the course literature (e.g., own notes, other literature on the topic, copies of slides, computers or calculators with the corresponding information stored). Only general aids are allowed, such as calculators without stored information of relevance to the course, general handbooks with formulae, and similar. 15 credits is required for a pass, maximum number of credits is 30. The results will be made public at latest Friday Nov. 17th, 2006.

Kindly observe:

- Always motivate your answers. An insufficient motivation can result in a deduction of credits. Conversely, also an erroneous answer can yield some credits, if it is clear from the motivation that the basic reasoning still is correct.
- Write legibly!
- Each sheet should be marked with name, personal number, number of assignment, and a running number for the sheets.
- Only one assignment per sheet.
- Please write only on one side of a sheet.
- The assignments are not necessarily sorted in order of difficulty. If you get stuck it might pay off to continue to the next assignment.
- Don't forget to fill in the course evaluation! You will find it at the homepage of the course → Course Evaluation → online form. If you need username and password, then mail me at `bjorn.lisper@mdh.se`.

Jour: Björn Lisper at 070-6729577.

---

---

## UPPGIFT 1 (4 POÄNG)

Write a function which takes a list of words, represented as strings, and return a list of the words where they are capitalized (First letter is a capital letter, and the rest of them are small letters). Full no. of credits requires that your solutions uses at least one of Haskell's builtin higher order functions in a meaningful way.

Note that your function must be able to handle empty strings. Non-alphabetic characters can be left as they are in the strings.

**Hint:** your solution may use Haskell's builtin functions `toUpper`, `toLower :: Char -> Char`, which convert to capital and small letter, respectively, and do not change non-alphabetic characters.

## UPPGIFT 2 (4 POÄNG)

Take your solution from the previous assignment and define an IO action that loops and reads a line from the terminal window into a string, converts the string into a list of words, transforms the list using your function, and converts back to a string and prints the result on the screen. If you haven't solved the previous assignment, then you can assume that you have a function and use that one. Your IO action must terminate when the empty string is input (i.e., when the user hits return directly).

**Hint:** you may use Haskell's builtin functions `words :: String -> [String]`, which converts a string into a list of strings of words, and `unwords :: [String] -> String` which converts a list of words back into a string.

## UPPGIFT 3 (3 POÄNG)

Consider the following function declarations:

```
repeat x = x : repeat x

select (x:xs) 0 = x
select (x:xs) n | n > 0 = select xs (n - 1)
                    | otherwise error "negative argument "
select [] n = error "empty list as argument "
```

What is the result of evaluating `select (repeat 17) 1` in Haskell? Motivate your answer, and describe how the evaluation proceeds.

**UPPGIFT 4** (4 POÄNG)

Define a function for exponentiation that calculates  $x^n$ , where  $n$  is a natural number, with a balanced multiplication tree. For instance,  $x^7$  is calculated as  $x^7 = x^3 \cdot x^4 = (x \cdot x^2) \cdot (x^2 \cdot x^2) = (x \cdot (x \cdot x)) \cdot ((x \cdot x) \cdot (x \cdot x))$ .

**Hint:** it might be good to use Haskell's builtin function `div` for integer division. For instance holds that `div 5 2 = 2`.

**UPPGIFT 5** (7 POÄNG)

Insertion sort is a classical sorting algorithm which you all already should know: it sorts a sequence of data by successively inserting a new datum from the unsorted sequence in the sorted one, until the unsorted sequence is empty and the sorted sequence contains all data from the unsorted sequence.

a) Implement a function for insertion sort in Haskell! Your function is supposed to sort list. (5p)

b) Which type will your function obtain, if there are no explicit typings? A thorough motivation is required to obtain full no. of credits. (2p)

**UPPGIFT 6** (5 POÄNG)

Number systems can be more or less primitive. One of the simplest ones imaginable has only three entities: "zero", "one", and "many". Nevertheless, such a number system might be useful in many situations, and one may want to represent it and use to calculate in different situations.

a) Define a data type in Haskell for this number system! (2p)

b) Make the data type an instance of the class `Num`. Choose reasonable implementations of the different methods in `Num`, and motivate why they are reasonable! In order to simplify the assignment, it is enough to give implementations of `+`, `negate`, and `fromInteger`. (3p)

**Hint:** the class declaration for `Num` looks as follows:

```
class (Eq a, Show a) => Num a where
  (+), (-), (*)      :: a -> a -> a
  negate            :: a -> a
  abs, signum       :: a -> a
  fromInteger       :: Integer -> a
```

**UPPGIFT 7** (3 POÄNG)

Beta-reduce the lambda expression  $(\lambda y.z) ((\lambda x.x x) x)$  in all possible orders, until it cannot be reduced further! For each step, show which redex is being reduced. What are the results (or the result)?

Good luck! Björn