COMP105 Lecture 11

Assignment 1

Outline

Today

- Assignment 1 overview

Relevant book chapters

- Programming In Haskell Chapter 3
- Learn You a Haskell Chapter 3

Assignment 1

Assignment 1 is **out now** – The handout is available on the website

A programming task in Haskell

- Designed to test your knowledge of recursion

Deadline: Wednesday the 18th of November (week 6) at 12:00 midday

Repeat encodings

If a character is repeated multiple times we can **repeat encode** it:

`"aaaa"` $\rightarrow$ `"a4"`
`"bb"` $\rightarrow$ `"b2"`
`"ccccc"` $\rightarrow$ `"c5"`

The repeat encoding

- deletes every repetition of the character
- replaces the repetitions with a number

Repeat encodings

To repeat encode a **string**, we replace all repeated characters with their repeat encodings:

`"aaaabbb"` $\to$ `"a4b3"`
`"aabbcc"` $\to$ `"a2b2c2"`
`"abc"` $\to$ `"abc"`
`"aaaaaaaaaaa"` $\to$ `"a11"`

Note that non-repeated characters are not repeat encoded

Simple repeat encoding

In the **simple repeat encoding**

- No character is ever repeated more than nine times.
- Single instances of a character will still be repeat encoded.

`"abc"` $\rightarrow$ `"a1b1c1"`
`"hello"` $\rightarrow$ `"h1e1l2o1"`

The following is not a valid simple repeat encoding: `"a11"`

The task

Part A (30%): build a decoder for the *simple* repeat encoding

- The functions you need to implement are given to you
- There are lots of hints

Part B (30%): build an encoder for the *simple* repeat encoding

- The functions you need to implement are given to you

Part C (40%): build an encoder and decoder for the full repeat encoding

- The function breakdown is not given to you

The task

You should use **recursion** wherever possible

- Q1 and Q4 do not need recursion, but all other questions do

To obtain full marks:

- Parts A and B: **do not use** any library functions except `head` and `tail`
- Part C: you may use `mod`, `div`, `length`, `reverse`, `head`, and `tail`
- You should not use list comprehensions

What is allowed?

The following are **allowed**

- Any operator that is infix by default.
  - Allowed: `+`, `++`, `&&`, `:`, etc.
  - Not allowed: `` `min` ``, `` `max` ``, etc.
- `if` expressions, `let` expressions, `where` syntax, guards.
- List ranges.
- Pattern matching.

The only things that are **not allowed** are

- Calling a library function (apart from the list of exceptions)
- List comprehensions

The template file

Fill out your solutions in the **template file** `Assignment1.hs`

It contains stub implementations for each question

```
char_to_int :: Char -> Integer
char_to_int = error "Not implemented"
```

- The first line is a **type annotation**
  - Do not modify it
- Replace the second line with your own code

Marking

You code will be tested by an **automated marker**

- Each function will be run on a number of test cases
- Code that works will get full marks

For non-working functions, a **human marker** can look at it

- Comment out the function
- Include the comment `-- PLEASE READ` above it

Coding Style

100% of the marks are for **correctness**

- No direct marks for coding style

However good coding style will help you develop the code

- So use comments
- Use `let` and `where`, guards, pattern matching

Commenting style

Advice on commenting

- Explain **what** the code is doing
- You can assume that the reader knows Haskell

```
-- This function adds its arguments together
add_two x y = x + y
```

This is far **too much**:

```
add_two -- here we give the function name
      x -- x is the first argument
      y -- y is the second argument
      = -- = is used to start the function body
      x + y -- we use the plus operator to add x to y
```

Coding style

Try to avoid writing all code on a **single** line

- Use `let`, `where`, pattern matching, and guards

```
remove_twos list = if length list == 0 then [] else
 (if head list == 2 then remove_twos (tail list) else
 head list : remove_twos (tail list))
```

```
remove_twos [] = []
remove_twos (x:xs)
    | x == 2    = rest
    | otherwise = x:rest
    where rest = remove_twos xs
```

Feedback

Feedback will be given as

- Personal report sent via email (test case results and comments)
- General feedback to the class in a lecture

Feedback should be given within **two weeks**

Submission

Save all your work into the template `Assignment1.hs`

Upload to the assessment task **submission system**

- `https://sam.csc.liv.ac.uk/COMP/Submissions.pl`

Automatic 5% **penalty** for submissions

- That do not compile (comment out code that does not compile)
- Where the type annotations have been modified

The checker

To **avoid** the 5% penalty, you can use the checker

- Download `Checker.hs` from the website
- Follow the instructions in the handout to load it

If the checker compiles successfully then you are **guaranteed** to not get the penalty

Deadline

The deadline is:

Wednesday the 18th of November (week 6) at 12:00 midday

Standard UoL **late penalty** applies:

- 5% deducted for each 24 hour period after the deadline
- No late submission after five 24 hour periods.

Extenuating circumstances (eg. illness, bereavement)

- Contact the student office ASAP
- Documentation will usually be required

Data integrity

It is your responsibility to ensure that your **data is safe**

Don't save all your work on a single USB drive!

Options:

- Maintain backup copies
- Use a cloud syncing service, eg. Dropbox or Google Drive
- Use git and Github/Bitbucket (not on public repos)

Academic integrity

The university takes **academic integrity** very seriously

Collusion

- Working together with another student
- You should **never** see another student's code

Plagiarism

- This is copying work from elsewhere
- Don't post the assignment on stack exchange...

Penalties for plagiarism can be severe

- At minimum, a zero mark for the assessment will be awarded

Homework schedule

Homework schedule:

- Week 4 – no homework
- Week 5 – homework sheet
- Week 6 – homework sheet

It might be best to get started on the assignment **this week**

- We will move on to other topics in the lectures
- You will not gain any benefit by waiting

Getting help

There is an **FAQ** at the end of the handout

Seek help if

- Your code does not compile and you don't know why
- You are really stuck on part A

Send me your code if it doesn't compile

- Please send the **entire file**
- I usually respond promptly (within reason...)

Don't leave it until Week 6!