CS331: Assignment #1

Due on Wednesday, January 05, 2022

Siddharth Charan 190101085

Assignment: Introduction

In this assignment we were asked to solve questions given in exercise of chapter 1 and chapter 2 of book "Programming in Haskell". There are 5 questions in 1st chapter and 5 are in second chapter, which are discussed below.

Question 1.1

In book 2nd solution which solves questions by solving outer variable, solves first double first. We are asked to give an alternate solution. So, we can solve outer double first and then we will solve second double first, hence our solution is different from solution given in book. Solution is given below:

double (double 2)
= { applying the outer double }
(double 2) + (double 2)
= { applying the second double }
(double 2) + (2 + 2)
= { applying the second + }
(double 2) + 4
= { applying the first double }
(2 + 2) + 4
= { applying the first + }
4 + 4
= { applying + }
8

Question 1.2

From book function sum is defined as below:
sum [] = 0 And
sum (n:ns) = n + sum ns

Hence using above definition, we can prove sum[x]=x like below:
sum [x]
= { applying sum }
x + sum []
= { applying sum }
x + 0
= { applying + }
x

Question 1.3

We can define product as below:
product [] = 1
product (n:ns) = n * product ns

Hence,
product [2,3,4]
= { applying product }
2 * (product [3,4])
= { applying product }
2 * (3 * product [4])
= { applying product }
2 * (3 * (4 * product []))
= { applying product }
2 * (3 * (4 * 1))
= { doing simple multiplication}
24

Question 1.4

In textbook definition for sorting(qsort) is already given , we will slightly change that definition and we will have achieve reverse sort function. Definition for qsort in book is same as below:
qsort[]=[]
qsort(x:xs) =qsort smaller ++ [x] ++ qsort larger
where,
smaller = {a|a<-xs, a<=x}
larger = b| b<-xs, b>x
(In this definition, ++ is an operator that appends two lists together.)
You can note that qsort is recursively defined and in this definition qsort is made of two qsort functions, and one element. If we see right side part, we can see that we have all elemens which are smaller than x, are in left side and greater elements are in right side. Hence, it is giving us a sorted output. But, if we swap locations of "qsort smaller" and "qsort larger" than, smaller elements will be in right side and greater elements will be in left side. So, new definition for qsort such that it gives us reverse sorted list.
qsort[]=[]
qsort(x:xs) =qsort larger ++ [x] ++ qsort smaller
where,
smaller = {a|a<-xs, a<=x}
larger = b| b<-xs, b>x
(In this definition, ++ is an operator that appends two lists together.)

Question 1.5

If we replace <= to < in definition of qsort. Then, for some lists which contains duplicate items, it won't be able to correctly sort the input list. Instead, of that as in smaller qsort part equal numbers are not there, hence those numbers will be removed. Hence, duplicate items will be removed from the list and we will get a sorted list as output without them. We can understand it with an example also.

Ex. qsort [2,2,3,1,1]
= { applying qsort }
qsort [1,1] ++ [2] ++ qsort [3]
= { applying qsort }
(qsort [] ++ [1] ++ qsort []) ++ [2]
++ (qsort [] ++ [3] ++ qsort [])
= { applying qsort, above property}
([] ++ [1] ++ []) ++ [2] ++ ([] ++ [3] ++ [])
= { applying ++ }
[1] ++ [2] ++ [3]
= { applying ++ }
[1,2,3]
(Note that duplicate 1s and 2s are removed with new definition.)

P.T.O.

Question 2.1

For this question code is given in separate file. But, screenshot of terminal is given below:

```
C:\jaipur>ghc op
[1 of 1] Compiling Main             ( op.hs, op.o )
Linking op.exe ...

C:\jaipur>op
"This is the question 1 part of the code:"
"double 2 = 4"
"factorial 10 = 3628800"
"average [1,2,3,4,5]3"
"quadruple 2 = 8"
"2+3*4 = 14"
"sqrt (3^2 + 4^2) = 5.0"
"head[1,2,3,4,5] = 1"
"tail [1,2,3,4,5] = [2,3,4,5]"
"[1,2,3,4,5] !! 2 = 3"
"take 3 [1,2,3,4,5] = [1,2,3]"
"drop 3 [1,2,3,4,5] = [4,5]"
"length [1,2,3,4,5] = 5"
"sum [1,2,3,4,5] = 15"
"product [1,2,3,4,5] = 120"
"[1,2,3] ++ [4,5] = [1,2,3,4,5]"
"reverse [1,2,3,4,5] = [5,4,3,2,1]"
"This is the question 4 part of the code:"
"For Question 4 we are taking input as [1,2,3,4,5] answer should be 5"
"using function1 we get"
5
"using function2 we get"
5
"Hence using both functions we are getting right answers."
"This is the question 5 part of the code:"
"For Question 5 we are taking input as [1,2,3,4,5] answer should be [1,2,3,4]"
"using function1 we get"
[1,2,3,4]
"using function2 we get"
[1,2,3,4]
"Hence using both functions we are getting right answers."
```

Question 2.2

Parenthesised expressions:

$(2 \wedge 3) * 4$

$(2 * 3) + (4 * 5)$

$2 + (3 * (4 \wedge 5))$

Question 2.3

3 mistakes:
1. N is given capital, it should be small(n).
2. Indentation for xs is wrong, it should be as par with above line.
3. We need to use backticks(‘’) instead of single quotes(‘’) to enclose div.

Corrected Script:

```
n = a `div` length xs
    where
        a = 10
        xs = [1,2,3,4,5]
```

And indeed these corrections make code working, you can check that in below screenshot.

```
n = a  `div` length xs
  where
     a = 10
     xs = [1,2,3,4,5]
main = do
     print n
```

```
C:\jaipur>ghc q3.hs
[1 of 1] Compiling Main
Linking q3.exe ...

C:\jaipur>q3
2
```

Question 2.4

For this question answer is given in separate file. Here, you can check the screenshot of terminal(attached in question 1) and also two functions which are used, I am writing here.
1. last1 xs = head (reverse xs)
2. last2 xs = head (drop (length xs - 1) xs)

Question 2.5

For this question answer is given in separate file. Here, you can check the screenshot of terminal(attached in question 1) and also two functions which are used, I am writing here.
1. init1 xs = reverse (tail (reverse xs))
2. init2 [] = error "list should not be empty"
   init2 [x] = []
   init2 (x:xs) = x : init2 xs