19CSE313-Principles of Programming Languages

Lab Exercise-1

Done By:

Adharsh S Mathew
CSE-D
AM.EN.U4CSE19302

Try the following Expressions

: 5+2

7

: 5 * 2 + 3

13

: sqrt 4.0

2.0

: sum [2,3,4]

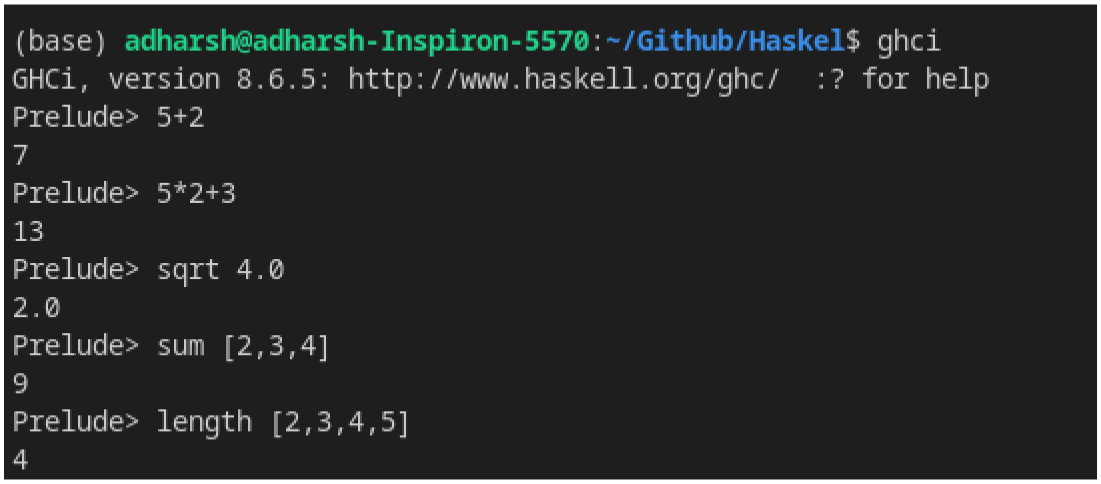
9

: length [2,3,4,5]

4

: sort [3,4,1,2,77,6]

[1, 2, 3, 4, 6, 77]



```
(base) adharsh@adharsh-Inspiron-5570:~/Github/Haskel$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> 5+2
7
Prelude> 5*2+3
13
Prelude> sqrt 4.0
2.0
Prelude> sum [2,3,4]
9
Prelude> length [2,3,4,5]
4
```

Writing your first script

The following exercises will give you practice in creating and loading script files.

Step 1

Create a file containing the following two lines. Each is a Haskell definition. Save the file as Ex1.hs

```
add x y = x + y
square x = x * x
```

Step 2

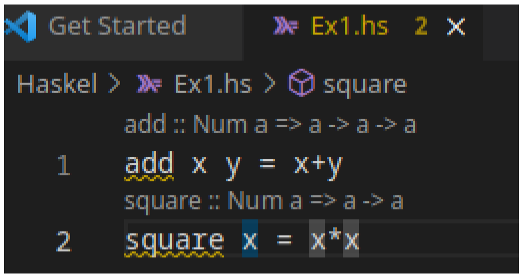
Until we load the new file, these definitions are not available for use. Try using these functions; you should get an error as shown below.

```
Prelude> add 3 + 4
ERROR: Undefined variable "add"
```

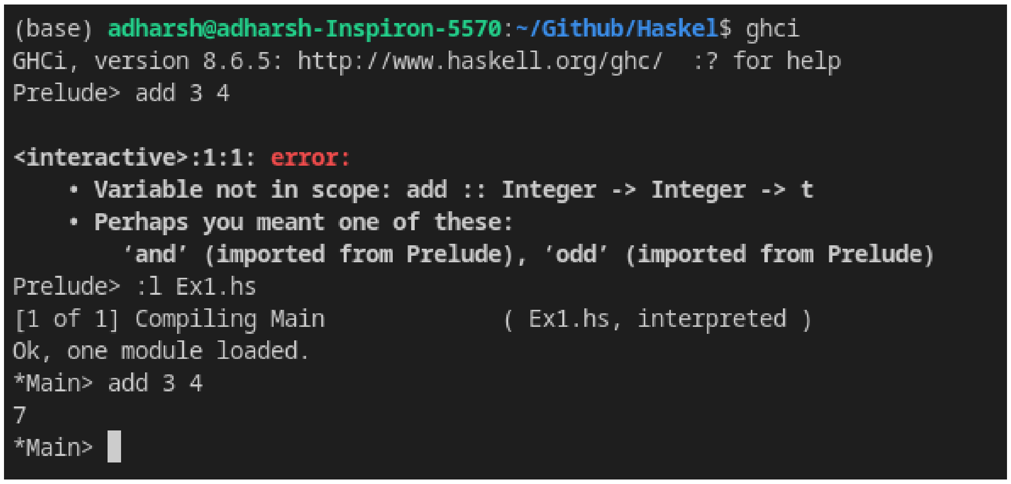
Step 3

Now load the new definitions using the load (:l) command and then try using the functions again.

```
Prelude> :l Ex1.hs
Reading file "Ex1.hs":

Hugs session for:
/home/rwatson/share/hugs/lib/Prelude.hs

Ex1.hs

Main> add 3 4
7
```



```
Get Started    Ex1.hs  2  ×
Haskel > Ex1.hs > square
add :: Num a => a -> a -> a
add x y = x+y
square :: Num a => a -> a
square x = x*x
```



```
(base) adharsh@adharsh-Inspiron-5570:~/Github/Haskel$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> add 3 4

<interactive>:1:1: error:
    • Variable not in scope: add :: Integer -> Integer -> t
    • Perhaps you meant one of these:
        ‘and’ (imported from Prelude), ‘odd’ (imported from Prelude)
Prelude> :l Ex1.hs
[1 of 1] Compiling Main             ( Ex1.hs, interpreted )
Ok, one module loaded.
*Main> add 3 4
7
*Main> 
```

Modules

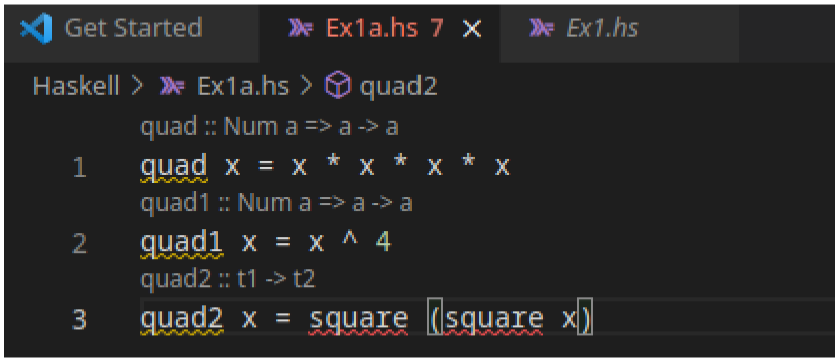
Hugs supports definitions spread over more than one file. In the following exercises we will create a second file and include it in the Hugs session.

Create a new file Ex1a.hs containing the new definitions:

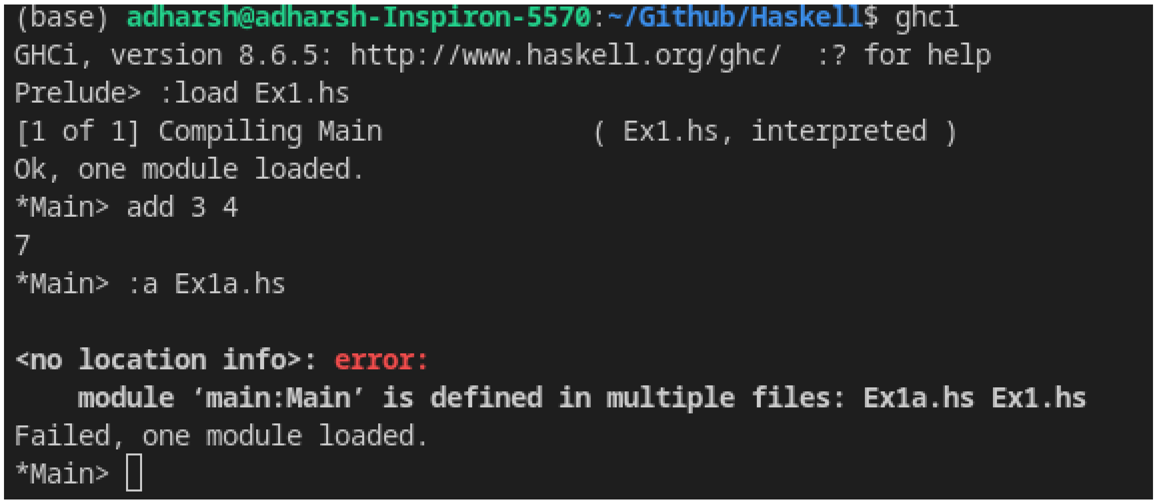
```
quad x = x * x * x * x
quad1 x = x ^ 4
quad2 x = square (square x)
```

and add it to the session with the command

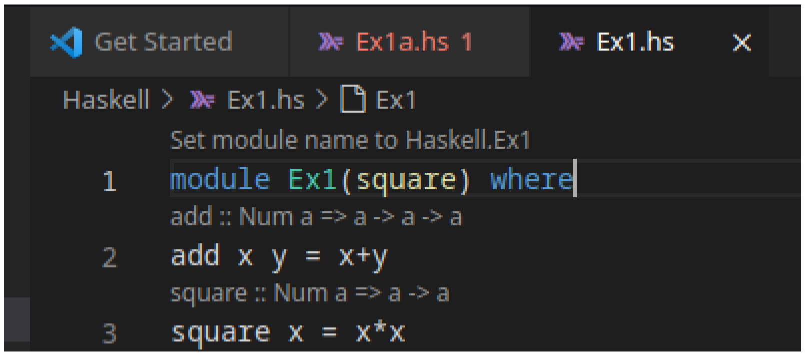
```
:a Ex1a.hs
```



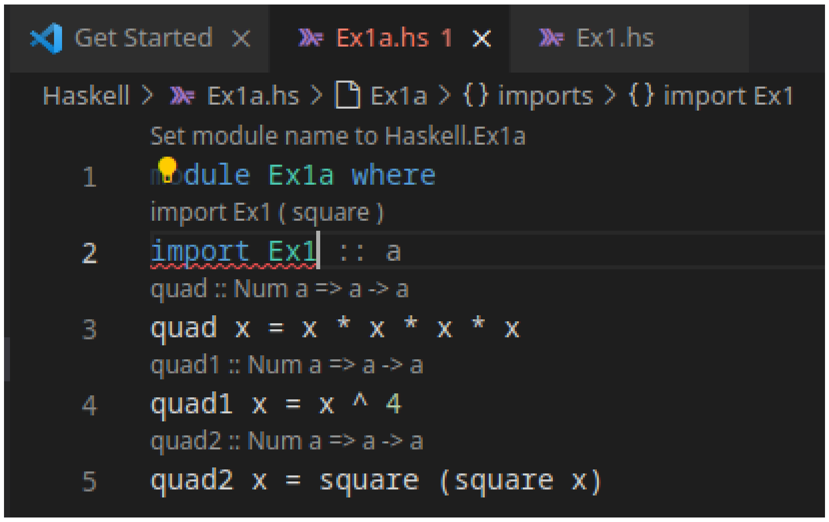
```haskell
quad :: Num a => a -> a
quad x = x * x * x * x
quad1 :: Num a => a -> a
quad1 x = x ^ 4
quad2 :: t1 -> t2
quad2 x = square (square x)
```



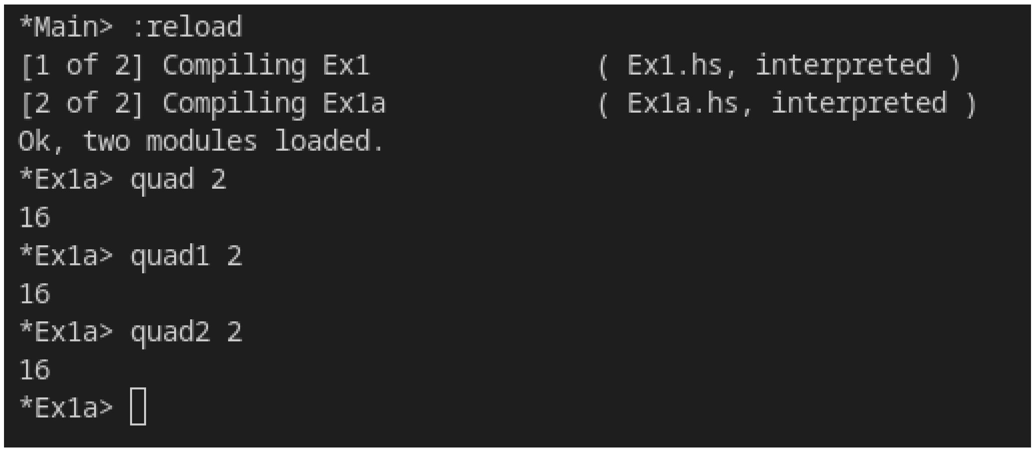
```
(base) adharsh@adharsh-Inspiron-5570:~/Github/Haskell$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> :load Ex1.hs
[1 of 1] Compiling Main             ( Ex1.hs, interpreted )
Ok, one module loaded.
*Main> add 3 4
7
*Main> :a Ex1a.hs

<no location info>: error:
    module ‘main:Main’ is defined in multiple files: Ex1a.hs Ex1.hs
Failed, one module loaded.
*Main> 
```



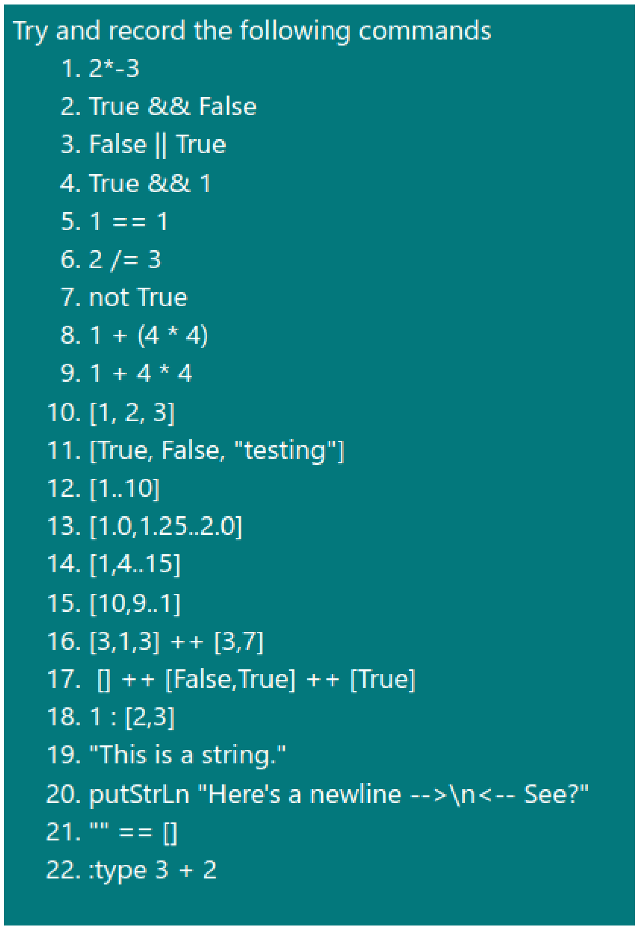
```
Set module name to Haskell.Ex1
module Ex1(square) where
add :: Num a => a -> a -> a
add x y = x+y
square :: Num a => a -> a
square x = x*x
```



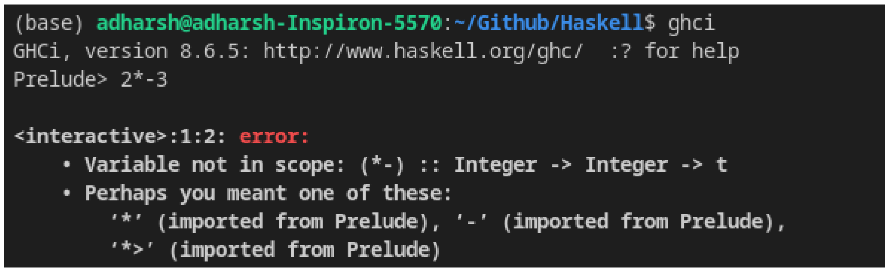
```
Set module name to Haskell.Ex1a
module Ex1a where
import Ex1 ( square )
import Ex1 :: a
quad :: Num a => a -> a
quad x = x * x * x * x
quad1 :: Num a => a -> a
quad1 x = x ^ 4
quad2 :: Num a => a -> a
quad2 x = square (square x)
```



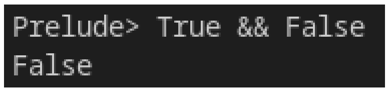
```
*Main> :reload
[1 of 2] Compiling Ex1              ( Ex1.hs, interpreted )
[2 of 2] Compiling Ex1a             ( Ex1a.hs, interpreted )
Ok, two modules loaded.
*Ex1a> quad 2
16
*Ex1a> quad1 2
16
*Ex1a> quad2 2
16
*Ex1a> 
```


Try and record the following commands

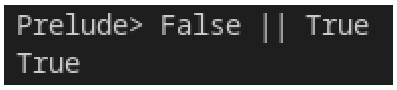
1. 2*-3
2. True && False
3. False || True
4. True && 1
5. 1 == 1
6. 2 /= 3
7. not True
8. 1 + (4 * 4)
9. 1 + 4 * 4
10. [1, 2, 3]
11. [True, False, "testing"]
12. [1..10]
13. [1.0,1.25..2.0]
14. [1,4..15]
15. [10,9..1]
16. [3,1,3] ++ [3,7]
17. [] ++ [False,True] ++ [True]
18. 1 : [2,3]
19. "This is a string."
20. putStrLn "Here's a newline -->\n<-- See?"
21. "" == []
22. :type 3 + 2

1. 

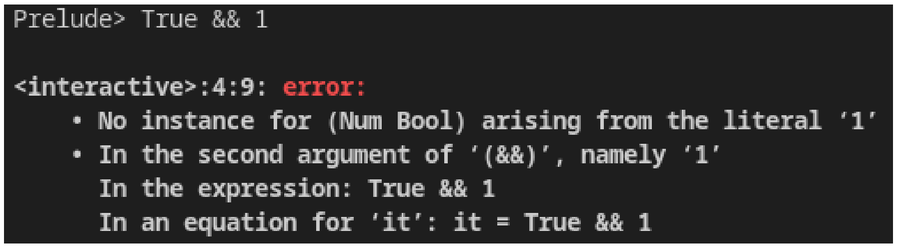
```
(base) adharsh@adharsh-Inspiron-5570:~/Github/Haskell$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> 2*-3

<interactive>:1:2: error:
    • Variable not in scope: (*-) :: Integer -> Integer -> t
    • Perhaps you meant one of these:
        ‘*’ (imported from Prelude), ‘-’ (imported from Prelude),
        ‘*>’ (imported from Prelude)
```

2. 

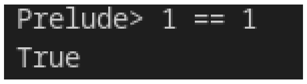
```
Prelude> True && False
False
```

3. 

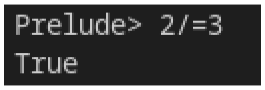
```
Prelude> False || True
True
```

4. 

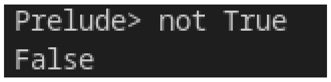
```
Prelude> True && 1

<interactive>:4:9: error:
    • No instance for (Num Bool) arising from the literal ‘1’
    • In the second argument of ‘(&&)’, namely ‘1’
      In the expression: True && 1
      In an equation for ‘it’: it = True && 1
```

5. 

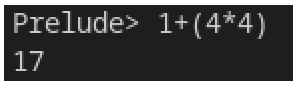
```
Prelude> 1 == 1
True
```

6. 

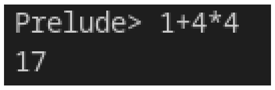
```
Prelude> 2/=3
True
```

7. 

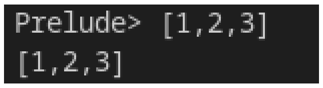
```
Prelude> not True
False
```

8. 

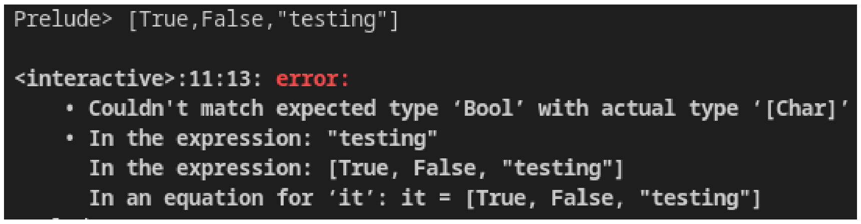
```
Prelude> 1+(4*4)
17
```

9. 

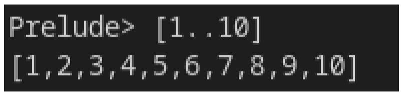
```
Prelude> 1+4*4
17
```

10. 

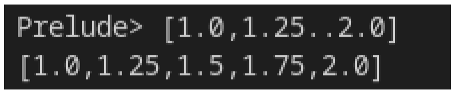
```
Prelude> [1,2,3]
[1,2,3]
```

11. 

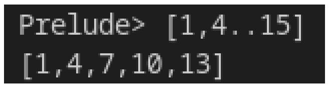
```
Prelude> [True,False,"testing"]

<interactive>:11:13: error:
    • Couldn't match expected type ‘Bool’ with actual type ‘[Char]’
    • In the expression: "testing"
      In the expression: [True, False, "testing"]
      In an equation for ‘it’: it = [True, False, "testing"]
```

12. 

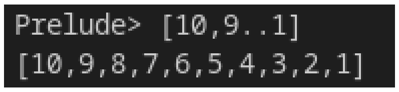
```
Prelude> [1..10]
[1,2,3,4,5,6,7,8,9,10]
```

13. 

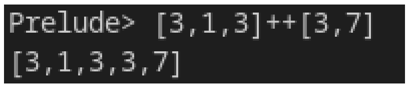
```
Prelude> [1.0,1.25..2.0]
[1.0,1.25,1.5,1.75,2.0]
```

14. 

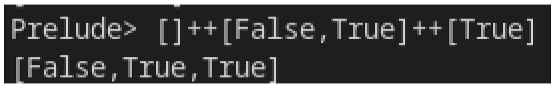
```
Prelude> [1,4..15]
[1,4,7,10,13]
```

15. 

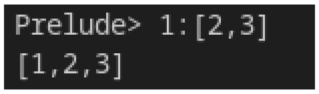
```
Prelude> [10,9..1]
[10,9,8,7,6,5,4,3,2,1]
```

16. 

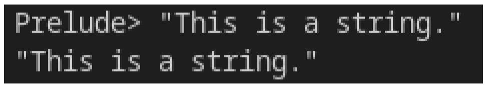
```
Prelude> [3,1,3]++[3,7]
[3,1,3,3,7]
```

17. 

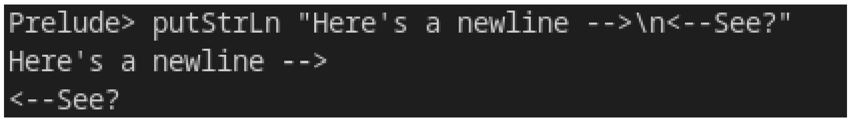
```
Prelude> []++[False,True]++[True]
[False,True,True]
```

18. 

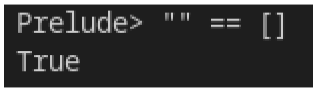
```
Prelude> 1:[2,3]
[1,2,3]
```

19. 

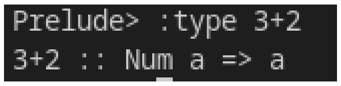
```
Prelude> "This is a string."
"This is a string."
```

20. 

```
Prelude> putStrLn "Here's a newline -->\n<--See?"
Here's a newline -->
<--See?
```

21. 

```
Prelude> "" == []
True
```

22. 

```
Prelude> :type 3+2
3+2 :: Num a => a
```