Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по ДЗ

Выполнил:
Студент группы ИУ5-33Б
Лупарев Сергей

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Постановка задачи

Разработать программу на языке программирования Haskell, реализующую постфиксный калькулятор.

Текст программы

calc.hs

```haskell
-- Пользовательский тип данных ошбики(либо слишком мало чисел, либо произошло деление на ноль)
data Error = StackTooShort Int Int | DividingByZero deriving Show

-- функция показа ошбики, выводит на экран ошибку с пояснением
showErr (StackTooShort a b) = "Stack has " ++ show a ++ " elements. " ++ show b ++ " needed"
showErr (DividingByZero) = "You divided by zero!"

-- Пользовательский тип данных токен, в который производится ввод и который может быть целым числом, либо действием между числами
data Token = Number Int | Plus | Minus | Div | Mul deriving Show

-- функция для извлечения значения из типа Maybe
numm (Just x) = x

-- data Either a b = Left a | Right b

-- функция, принимающая список чисел, которая возвращает список со сложенными первыми двумя числами(в случае нехватки чисел возвращает ошибку)
add [] = Left (StackTooShort 0 2)
add (x:[]) = Left (StackTooShort 1 2)
add (x1:x2:xs) = Right ((x1+x2):xs)

-- Вычитание. Аналогично add
sub [] = Left (StackTooShort 0 2)
sub (x:[]) = Left (StackTooShort 1 2)
sub (x1:x2:xs) = Right ((x1-x2):xs)
--sub (x1:x2:xs) = Right ((x2 - x1):xs)

-- Целочисленное деление. Аналогично add
div' [] = Left (StackTooShort 0 2)
div' (x:[]) = Left (StackTooShort 1 2)
div' (x1:x2:xs) = if x2/=0 then Right ((div x1 x2):xs) else Left DividingByZero
--div' (x1:x2:xs) = if x2/=0 then Right ((div x2 x1):xs) else Left DividingByZero

-- Деление. Аналогично add
divide [] = Left (StackTooShort 0 2)
divide (x:[]) = Left (StackTooShort 1 2)
divide (x1:x2:xs) = if x2/=0 then Right ((x1/x2):xs) else Left DividingByZero

-- Умножение. Аналогично add
mul [] = Left (StackTooShort 0 2)
mul (x:[]) = Left (StackTooShort 1 2)
mul (x1:x2:xs) = Right ((x1*x2):xs)

-- функция преобразования токена и списка целых чисел в список чисел
-- Если внутри токена оператор, то он выполняется
-- Если внутри токена число, то оно присоединяется к списку
operation :: Token -> [Int] -> Either Error [Int]
operation Plus a = add a
operation Minus a = sub a
operation Div a = div' a
operation Mul a = mul a
operation (Number a) b = Right (a:b)

```

```haskell
-- функция преобразования строки в число
parseInt :: String -> Maybe Int
parseInt str = h (reads str) -- reads разбивает строку на пары
  where
    -- h :: [(Int, String)] -> Maybe Int
    h [(a, b)] = if b=="" then Just a else Nothing
    h _ = Nothing

-- функция, разбивающая входную строку на слова и преобразующая их в токен
convert a = convert' (words a)
  where
    convert' [] = []
    convert' (x:xs) = (if (x=="+") then Plus else if (x=="-") then Minus else if (x=="/") then Div else if (x=="*") then Mul else Number (numm (parseInt x))) : (convert' xs)

-- функция, которая принимает стек целых чисел и список токенов, а затем выполняет операции, представленные токенами, на значениях стека
calc :: [Int] -> [Token] -> Either Error [Int]
calc stack [] = Right stack
calc stack (op:ops) = do
    stack <- operation op stack
    calc stack ops

-- функция вывода результата
ret:: Either Error [Int] -> String
ret (Left a) = showErr a
ret (Right (x:xs)) = show x


--prl:: [Token] -> [Int]
--prl ((Number x) : xs) = x : prl xs
--prl (x) = []

-- функция, которая принимает список токенов и возвращает новый список токенов, содержащий только цифровые токены (числа)
prl:: [Token] -> [Token]
prl ((Number x) : xs) = (Number x) : (prl xs)
prl (x) = []

-- функция, которая принимает список токенов и возвращает список токенов, содержащий только операции
pr2:: [Token] -> [Token]
pr2 (Plus:xs) = Plus : pr2 xs
pr2 (Minus:xs) = Minus : pr2 xs
pr2 (Div:xs) = Div : pr2 xs
pr2 (Mul:xs) = Mul : pr2 xs
pr2 (x:xs) = pr2 xs
pr2 [] = []

-- функция, переворачивающая список
reverseList [] = []
reverseList (x:xs) = reverseList xs ++ [x]
```

```haskell


--runProgram a = (ret (calc [] (convert a)))


-- Запуск программы (преобразуем все слова в токены, разбиваем токены на числа и операции раздельно, а затем переворачиваем список, чтобы операции оказались сверху)
runProgram a = (ret (calc [] ( (reverseList (prl (convert a))) ++ ( pr2 (convert a) ) )))

-- Ввод данных
interact' :: (String -> String) -> IO ()
interact' f = do
    a <- getLine
    putStrLn (f a)
    interact' f

-- Запуск
main = interact' runProgram
```

Результат выполнения программы