ISTANBUL TECHNICAL UNIVERSITY

BLG 458-E
FUNCTIONAL PROGRAMMING

PROJECT REPORT

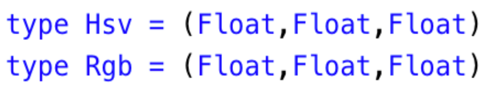
EMRE UYSAL
150160510

TABLE OF CONTENTS

*1.*QUESTION 1

1.a. Information

In this question, we are required to implement a function that converts RGB triple to HSV triple. So, in this step I created types for RGB and HSV. Basically they are same types (float,float,float) but I separated them because of making easier to read the code. I used float because we need to find % values in HSV. For S and V part I represented the values between 0 and 1.



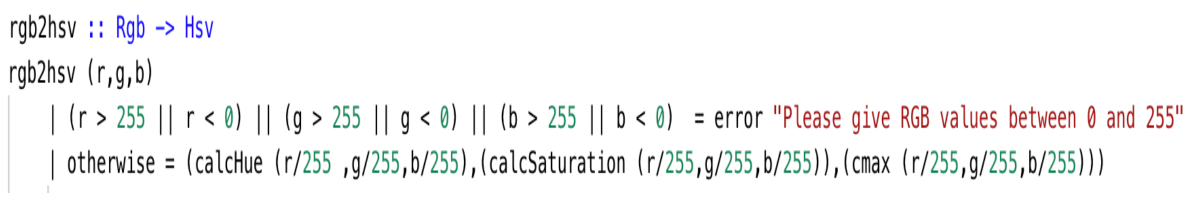
```haskell
type Hsv = (Float,Float,Float)
type Rgb = (Float,Float,Float)
```

Picture 1 Types for RGB and HSV

1.b. Implementation

First, I created rgb2hsv function that takes type RGB and returns HSV then I checked the RGB values if they are between 0 and 255. I provided meaningful error messages using error function. I created functions according to given mathematical formula. So, I implemented cmax, cmin, delta, calcSaturation, calcHue helper functions.

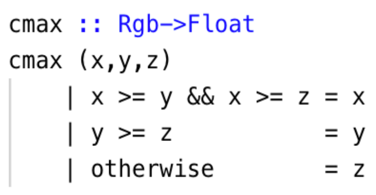
1.b.1. rgb2hsv



```haskell
rgb2hsv :: Rgb -> Hsv
rgb2hsv (r,g,b)
    | (r > 255 || r < 0) || (g > 255 || g < 0) || (b > 255 || b < 0)  = error "Please give RGB values between 0 and 255"
    | otherwise = (calcHue (r/255 ,g/255,b/255),(calcSaturation (r/255,g/255,b/255)),(cmax (r/255,g/255,b/255)))
```

Function takes RGB and returns HSV. First it checks r,g,b values then calls helper functions to calculate HSV.

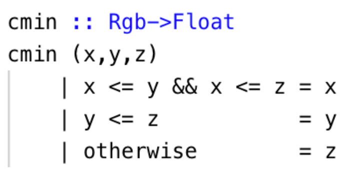
1.b.2. cmax



```haskell
cmax :: Rgb->Float
cmax (x,y,z)
    | x >= y && x >= z = x
    | y >= z           = y
    | otherwise        = z
```

This function takes RGB and returns the maximum value as float.

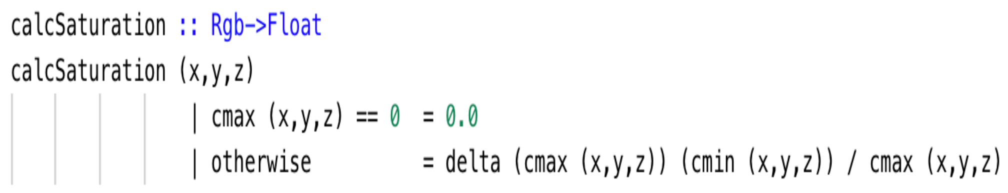
1.b.3. cmin



```haskell
cmin :: Rgb->Float
cmin (x,y,z)
    | x <= y && x <= z = x
    | y <= z           = y
    | otherwise        = z
```

This function takes an RGB and returns minimum value as float.

1.b.4. calcSaturation

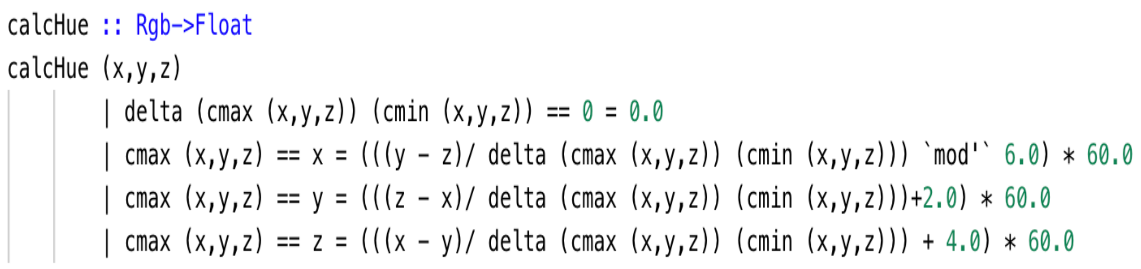


```haskell
calcSaturation :: Rgb->Float
calcSaturation (x,y,z)
                | cmax (x,y,z) == 0  = 0.0
                | otherwise          = delta (cmax (x,y,z)) (cmin (x,y,z)) / cmax (x,y,z)
```

This function takes an RGB and returns a float. It makes calculations according to this function;

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

1.b.5. calcHue

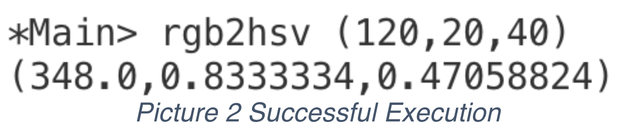


```haskell
calcHue :: Rgb->Float
calcHue (x,y,z)
        | delta (cmax (x,y,z)) (cmin (x,y,z)) == 0 = 0.0
        | cmax (x,y,z) == x = (((y - z)/ delta (cmax (x,y,z)) (cmin (x,y,z))) `mod'` 6.0) * 60.0
        | cmax (x,y,z) == y = (((z - x)/ delta (cmax (x,y,z)) (cmin (x,y,z)))+2.0) * 60.0
        | cmax (x,y,z) == z = (((x - y)/ delta (cmax (x,y,z)) (cmin (x,y,z))) + 4.0) * 60.0
```

This function takes an RGB and returns a float number. It makes calculations according to this function;

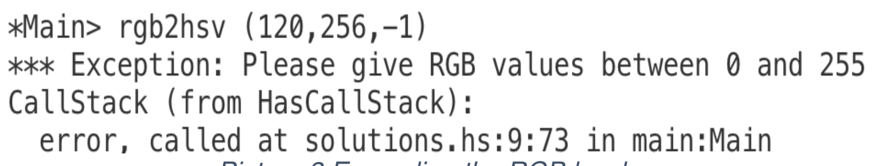
$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

1.c. Execution



```
*Main> rgb2hsv (120,20,40)
(348.0,0.8333334,0.47058824)
```

Picture 2 Successful Execution



```
*Main> rgb2hsv (120,256,-1)
*** Exception: Please give RGB values between 0 and 255
CallStack (from HasCallStack):
  error, called at solutions.hs:9:73 in main:Main
```

Picture 3 Exceeding the RGB borders

2.QUESTION 2

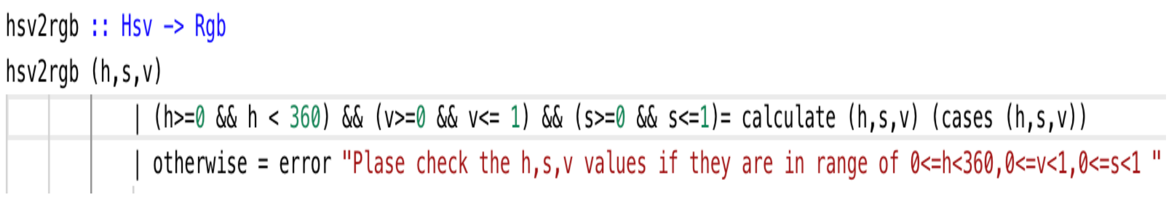
2.a. Information

In this question, we are required to write a function that converts HSV triple to RGB triple. Again, I used the RGB and HSV types that I created before.

2.b. Implementation

I created hsv2rgb function which takes HSV and returns RGB. First it checks HSV values if they are exceeding the borders or not. Then it makes calculation according to given mathematical formula. I used c,m,x,cases and calculate helper functions to reach the result.

2.b.1. hsv2rgb



```haskell
hsv2rgb :: Hsv -> Rgb
hsv2rgb (h,s,v)
        | (h>=0 && h < 360) && (v>=0 && v<= 1) && (s>=0 && s<=1)= calculate (h,s,v) (cases (h,s,v))
        | otherwise = error "Plase check the h,s,v values if they are in range of 0<=h<360,0<=v<1,0<=s<1 "
```

This function takes HSV and returns RGB. First it checks the borders for HSV. According to result it makes calculation or gives an error.

2.b.2. c

```haskell
c :: Hsv -> Float
c (h,s,v) = v*s
```

This function takes HSV and returns s*v.

2.b.3. x

```haskell
x :: Hsv -> Float
x (h,s,v) = (c (h,s,v)) * (1- abs(((h/60) `mod'` 2)-1))
```

This function takes HSV and returns the calculation according to given formula;

$$X = C \times (1 - |(H / 60°) \bmod 2 - 1|)$$

2.b.4. m

```haskell
m :: Hsv -> Float
m (h,s,v)= v - (c (h,s,v))
```

This function takes HSV and returns v-(v*s).

2.b.5. cases

```
cases :: Hsv -> Rgb
cases (h,s,v)
    | (h>=0   && h < 60) = (c (h,s,v),x (h,s,v),0)
    | (h>=60  && h < 120) = (x (h,s,v),c (h,s,v),0)
    | (h>=120  && h < 180) = (0,c (h,s,v),x (h,s,v))
    | (h>=180  && h < 240) = (0,x (h,s,v),c (h,s,v))
    | (h>=240  && h < 300) = (x (h,s,v),0,c (h,s,v))
    | (h>=300  && h < 360) = (c (h,s,v),0,x (h,s,v))
```

This function takes HSV and returns the matched case according to given formula;

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0^\circ \leq H < 60^\circ \\ (X, C, 0) & , 60^\circ \leq H < 120^\circ \\ (0, C, X) & , 120^\circ \leq H < 180^\circ \\ (0, X, C) & , 180^\circ \leq H < 240^\circ \\ (X, 0, C) & , 240^\circ \leq H < 300^\circ \\ (C, 0, X) & , 300^\circ \leq H < 360^\circ \end{cases}$$

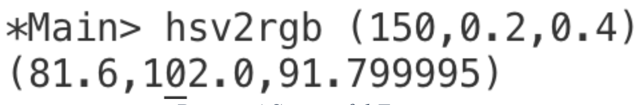
2.b.6. calculate

```
calculate ::Hsv -> Rgb -> Rgb
calculate (h,s,v) (r,g,b) = ((r+(m (h,s,v)))*255,(g+(m (h,s,v)))*255,(b+(m (h,s,v)))*255)
```

This function takes HSV and RGB and makes calculations according to given formula;

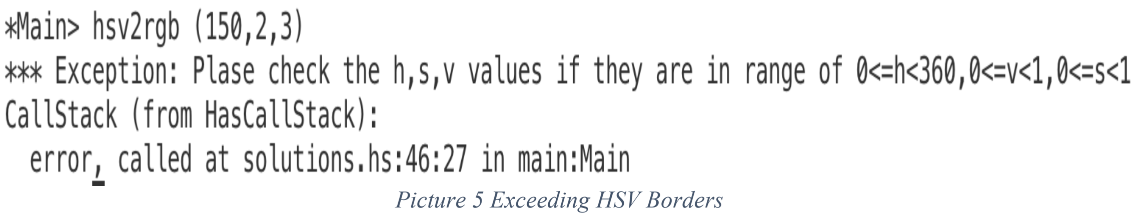
$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

2.c. Execution



```
*Main> hsv2rgb (150,0.2,0.4)
(81.6,102.0,91.799995)
```

Picture 4 Successful Execution



```
*Main> hsv2rgb (150,2,3)
*** Exception: Plase check the h,s,v values if they are in range of 0<=h<360,0<=v<1,0<=s<1
CallStack (from HasCallStack):
  error, called at solutions.hs:46:27 in main:Main
```

Picture 5 Exceeding HSV Borders

3. QUESTION 3

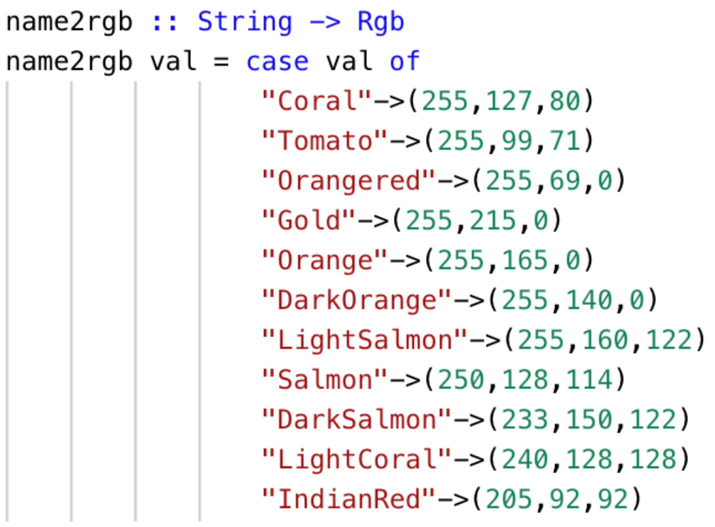
3.a. Information

In this question, we are required to write a function that converts html color name to RGB. I searched for a relation between color names and RGB values, but I could not find any relation between them. So, I need to give matchings for RGB values and names to program.

3.b. Implementation

When I writing name2rgb function, I preferred the case of capability of Haskell. I write the all html color names and RGB values.

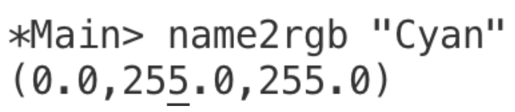
3.b.1. name2rgb



```haskell
name2rgb :: String -> Rgb
name2rgb val = case val of
                "Coral"->(255,127,80)
                "Tomato"->(255,99,71)
                "Orangered"->(255,69,0)
                "Gold"->(255,215,0)
                "Orange"->(255,165,0)
                "DarkOrange"->(255,140,0)
                "LightSalmon"->(255,160,122)
                "Salmon"->(250,128,114)
                "DarkSalmon"->(233,150,122)
                "LightCoral"->(240,128,128)
                "IndianRed"->(205,92,92)
```

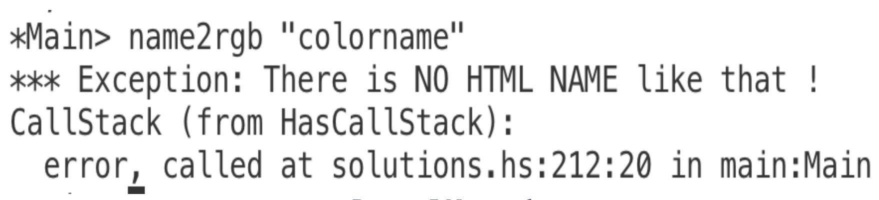
This function takes a String and returns a RGB result using case of. It matches the given string with already written one and returns result. But it has disadvantage of being case sensitive. So, I took the W3 HTML color names as a reference.

3.c. Execution



```
*Main> name2rgb "Cyan"
(0.0,255.0,255.0)
```

Picture 6 Match with Cyan



```
*Main> name2rgb "colorname"
*** Exception: There is NO HTML NAME like that !
CallStack (from HasCallStack):
  error, called at solutions.hs:212:20 in main:Main
```

Picture 7 No match

4. QUESTION 4

4.a. Information

In this question, we need to find html color gradients by using linear interpolation. So, we need to take a starting HSV, ending HSV and step number.

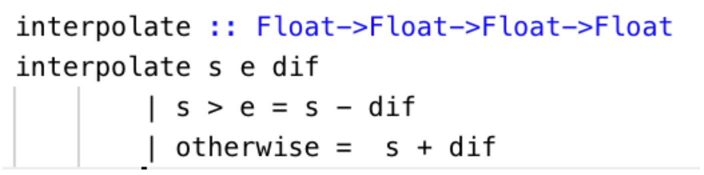
4.b. Implementation

In this question, we need to find difference between starting and ending values and divide them to given step number. After that, we need increase or decrease the values according to range.

4.b.1. hsvGradient

This function takes 2 HSV values and a step number. It checks the ranges of HSV values and adds the starting HSV to a list. Then calls itself and recursively by decreasing step number 1, calculates the gradient by using interpolate function. (I could not give the code screenshot here because, it exceeds the page borders.)

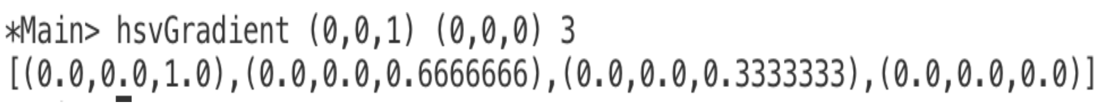
4.b.2. interpolate



```
interpolate :: Float->Float->Float->Float
interpolate s e dif
          | s > e = s - dif
          | otherwise =  s + dif
```

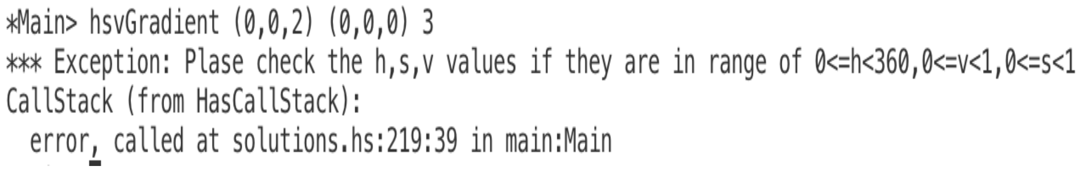
Interpolate function takes starting, ending and difference values. It makes comparison between starting and ending values and gives the result as added or subtracted by difference. All other steps of interpolation are included in hsvGradient function when calling itself.

4.c. Execution



```
*Main> hsvGradient (0,0,1) (0,0,0) 3
[(0.0,0.0,1.0),(0.0,0.0,0.6666666),(0.0,0.0,0.3333333),(0.0,0.0,0.0)]
```

Picture 8 hsvGradient Successful



```
*Main> hsvGradient (0,0,2) (0,0,0) 3
*** Exception: Plase check the h,s,v values if they are in range of 0<=h<360,0<=v<1,0<=s<1
CallStack (from HasCallStack):
  error, called at solutions.hs:219:39 in main:Main
```

Picture 9 hsvGradient Exception

*5.*QUESTION 5

5.a. Information

In this question, we are required to provide a description for given HSV triple. The description is String. So, function must get HSV and return a String.

5.b. Implementation

According to reference given in the project documentation, we need to calculate ranges and matching definition to that ranges. So, I defined the range for HSL values, because it is more easy to understand from the color ranges and converted the HSV triple to HSL.

5.b.1. hsv2desc

```
hsv2desc :: Hsv -> String
hsv2desc (h,s,v)
         | (h>=0 && h < 360) && (v>=0 && v<= 1) && (s>=0 && s<=1) =  (hue h) ++ "," ++ (sat(calcS)) ++ " " ++ (light(calcL))
         | otherwise = error "Plase check the h,s,v values if they are in range of 0<=h<360,0<=v<1,0<=s<1 "
```

This function checks the HSV triple and calculates hue, saturation and light values and returns a string.

5.b.2. calcL and calcS

```
calcL = (2-s) * v / 2
calcS = (s*v) / (if calcL < 0.5 then calcL*2 else 2-calcL*2)
```

These function converts HSV to HSL, so they calculate the S an L values.

5.b.3. hue

```
hue :: Float -> String
hue val
    | val>0 && val <= 15       ="red"
    | val >  15 && val <= 45  = "orange"
    | val >  45 && val <= 70  = "yellow"
    | val >  70 && val <= 79  = "lime"
    | val >  79 && val <= 163 = "green"
    | val > 163 && val <= 193 = "cyan"
    | val > 193 && val <= 240 = "blue"
    | val > 240 && val <= 260 = "indigo"
    | val > 260 && val <= 270 = "violet"
    | val > 270 && val <= 291 = "purple"
    | val > 291 && val <= 327 = "magenta"
    | val > 327 && val <= 244 = "rose"
    | val > 344 && val <= 360 = "red"
```

This function find matched range for given hue value and returns a string.

5.b.4. sat

```
sat :: Float -> String
sat val
    | val > 0.9                  = "very saturated"
    | val > 0.8                  = "rather saturated"
    | val > 0.6                  = "saturated"
    | val > 0.46                 = "rather unsaturated"
    | val > 0.3                  = "unsaturated"
    | val > 0.1                  = "very unsaturated"
    | val > 0.03                 = "almost grey"
    | val <= 0.03                = "grey"
    | otherwise                  = "grey"
```

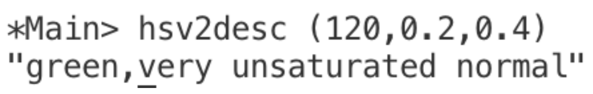
This function find matched range for given saturation value and returns a string.

5.b.5. light

```
light :: Float -> String
light val
    | val > 0.94                 = "almost white"
    | val > 0.8                  = "very light"
    | val > 0.6                  = "light"
    | val > 0.3                  = "normal"
    | val > 0.22                 = "dark"
    | val < 0.1 && val > 0.09    = "very dark"
    | val >= 0.0 && val <= 0.09 = "almost black"
    | otherwise                  = "black"
```

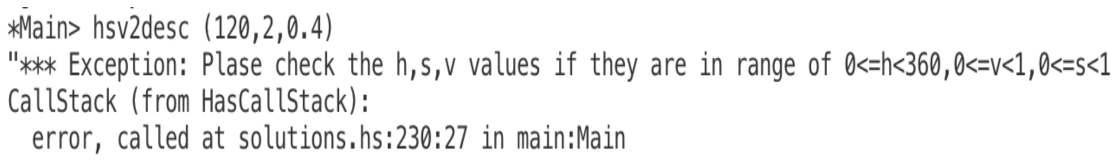
This function find matched range for given lightness value and returns a string.

5.c. Execution



```
*Main> hsv2desc (120,0.2,0.4)
"green,very unsaturated normal"
```

Picture 10 Successful Execution



```
*Main> hsv2desc (120,2,0.4)
"*** Exception: Plase check the h,s,v values if they are in range of 0<=h<360,0<=v<1,0<=s<1
CallStack (from HasCallStack):
  error, called at solutions.hs:230:27 in main:Main
```

Picture 11 Exception

6. QUESTION 6

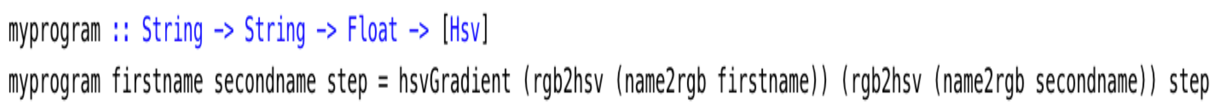
6.a. Introduction

In this question we are required to implement a function that takes a starting and ending html names and a step number. The function must produce gradient list between them.

6.b. Implementation

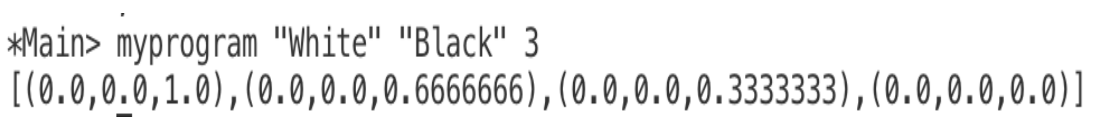
I used functions; name2rgb to convert given name to html color name, rgb2hsv to convert RGB to HSV triple and hsvGradient to calculate gradients.

6.b.1. myprogram



```haskell
myprogram :: String -> String -> Float -> [Hsv]
myprogram firstname secondname step = hsvGradient (rgb2hsv (name2rgb firstname)) (rgb2hsv (name2rgb secondname)) step
```

6.c. Execution



```
*Main> myprogram "White" "Black" 3
[(0.0,0.0,1.0),(0.0,0.0,0.6666666),(0.0,0.0,0.3333333),(0.0,0.0,0.0)]
```

Picture 12 Result of myprogram