Haskell Hwk1

Martin Kozeny
CSCI 4501: Programming Language Structure
Spring 2011 University of New Orleans

February 9, 2011

1 Haskell code

```
import Char(isLower,toUpper,toLower)

-- this function count area of triangle clasiclly s= (a*v)/2
-- input parameters are Floats which are >0 (there is a control if parameters are > 0)
-- and return value has also type Float
triangleArea :: Float -> Float -> Float
triangleArea a v = if ((a <= 0) || (v <= 0))
                   then error "Base and height have to be greater than 0"
                   else (a * v) / 2

-- this function  change the letter case
-- as an input char is expected and according to his case will be this letter changed
-- to another case
-- if is in lower case, than is changed to upper case, otherwise is expected as char
-- in upper case and changed to lower case
changeLetterCase :: Char -> Char
changeLetterCase a = if isLower a
                     then toUpper a
                     else toLower a

-- as input is expected integer from 0 to 100
-- returned value is String according to input number, if is lower than 0 or greater
-- than 100, an error is thrown
letterGrade :: Int -> String
letterGrade grade
     | grade >= 90 && grade <= 100 = "A"
     | grade >= 80 && grade <= 89 = "B"
     | grade >= 70 && grade <= 79 = "C"
     | grade >= 60 && grade <= 69 = "D"
     | grade >= 0 && grade <= 59 = "E / F"
     | otherwise = error "Bad numeric grade"

-- as an input two integers >0(there is a control if parameters are > 0) are expected
-- and returned value is also integer
-- if first number modulo second number is nonzero, then the power is zero
-- otherwise is first number modulo second number is zero, the power is at least 1
-- so the result is 1 + and recursively count the power for first number
-- and for second number divided by first number
```

```
highestPowerDivisor :: Int -> Int -> Int
highestPowerDivisor number1 number2
     | number1 <= 0 || number2 <= 0 = error "Input parameters must > 0"
     | mod number2 number1 /= 0 = 0
     | mod number2 number1 == 0 = 1 + highestPowerDivisor number1 (div number2 number1)

-- as an input 3 integers are expected
-- return value is also integer
-- if first argument is greater than second argument or third argument lower than zero,
-- an error is thrown
-- if first argument plus third argument are higher than second argument,
-- function returns first argument because second argument is border of range
-- else if first argument plus third argument are equal to second argument, function
-- returns first argument plus second argument
-- (it is also possible return twice first argument plus third argument)
-- else if first plus third argument is lower than second argument,
-- we add first argument to recursion call where first argument is previous first argument
-- plus added range ( = third argument ), other parameters are same
addRange :: Int -> Int -> Int -> Int
addRange a b c
    | a > b = error "First parameter must be <= than second parameter"
    | c < 0 = error "Third parameter must be > than 0"
    | a + c > b = a
    | a + c == b = a + b
    | a + c < b = a + addRange (a + c) b c

-- this function expects three Floats which are length of concrete side of triangle
-- and returns value which is also float
-- firstly it is tested if values of sides describe a trinagle
-- if yes, area of triangle is counted using parameter 's' counted in 'where' clause
areaTriangle :: Float  -> Float -> Float -> Float
areaTriangle a b c
        | a <= 0 || b <= 0 || c <= 0 = error "Every side must be > 0"
        | (a + b <= c) || (a + c <= b) || (b + c <= a) =
                error "Sides a, b, c do not describe a triangle"
        |otherwise = sqrt(s*(s-a)*(s-b)*(s-c))
        where s= (a+b+c)/2

-- this function as previous function expects three Floats which are length
-- of concrete side of triangle and returns value which is also float
-- also here it is firstly tested if values of sides describe a trinagle
-- if yes, area of triangle is counted in 'in' clause using parameter 's'
-- counted in 'let' clause
areaTriangle' :: Float -> Float -> Float -> Float
areaTriangle' a b c
     | a <= 0 || b <= 0 || c <= 0 = error "Every side must be > 0"
     |(a + b <= c) || (a + c <= b) || (b + c <= a) =
                error "Sides a, b, c do not describe a triangle"
     |otherwise = let s=(a+b+c)/2
     in  sqrt(s*(s-a)*(s-b)*(s-c))
```

2 Test script

```
-- 1.
triangleArea 5 3
-- expected 7.5
triangleArea 4 2
-- expected 4.0
triangleArea 7 8
-- expected 28.0
triangleArea 14 5
-- expected 35.0
triangleArea (-3) 7
-- expected error 'Base and height have to be greater than 0'

-- 2.
changeLetterCase 'c'
-- expected 'C'
changeLetterCase 'A'
-- expected 'a'
changeLetterCase 'f'
-- expected 'F'
changeLetterCase 'G'
-- expected 'g'
changeLetterCase 'w'
-- expected 'W'
changeLetterCase 'X'
-- expected 'x'

-- 3.
letterGrade 95
-- expected 'A'
letterGrade 90
-- expected 'A'
letterGrade 65
-- expected 'D'
letterGrade 50
-- expected 'D'
letterGrade 47
-- expected 'E / F'
letterGrade 150
-- expected error 'Bad numeric grade'
letterGrade (-3)
-- expected 'Bad numeric grade'

-- 4.
highestPowerDivisor 2 50
-- expected 1
highestPowerDivisor 5 250
-- expected 3
highestPowerDivisor 25 10
-- expected 0
highestPowerDivisor 7 100
-- expected 0
highestPowerDivisor (-2) 13
-- expected error 'Input parameters must > 0'
```

```
-- 5.
addRange 0 5 1
-- expected 15
addRange 1 20 3
-- expected 70
addRange 20 50 4
-- expected 272
addRange 30 20 3
-- expected error 'First parameter must be     than second parameter'
addRange 10 20 (-2)
-- expected error 'Third parameter must be > than 0'

-- 6.1.
areaTriangle 3 4 6
-- expected 5.332682
areaTriangle 3 4 2
-- expected 2.9047375
areaTriangle 2 7 8
-- expected 6.4371967
areaTriangle 14 15 12
-- expected 78.92679
areaTriangle 7 15 7
-- expected error 'Sides a, b, c do    describe a triangle'

-- 6.2.
areaTriangle' 3 4 6
-- expected 5.332682
areaTriangle' 3 4 2
-- expected 2.9047375
areaTriangle' 2 7 8
-- expected 6.4371967
areaTriangle' 14 15 12
-- expected 78.92679
areaTriangle' 7 15 7
-- expected error 'Sides a, b, c do    describe a triangle'
```

3 C code

```
/*libraries*/
#include <ctype.h>
#include <math.h>


/*headers of functions*/
float triangleArea(float a, float v);
char changeCase(char a);
char * letterGrade(int grade);
int highestPowerDivisor(int number1, int number2);
int addRange(int a, int b, int c);
double areaTriangle(double a, double b, double c);
/*testing functions is in main thread*/
int main()
{
    /*TESTING OF TRIANGLE AREA*/
    printf("\r\nTESTING OF TRIANGLE AREA\r\n");
    float triangleA = triangleArea(5,3);
    if(triangleA != -1)
        printf ("1. Triangle area, where base is 5 and height is 3, is %f\r\n",triangleA);
    triangleA = triangleArea(4,2);
    if(triangleA != -1)
        printf ("1. Triangle area, where base is 4 and height is 2, is %f\r\n",triangleA);
    triangleA = triangleArea(7,8);
    if(triangleA != -1)
        printf ("1. Triangle area, where base is 7 and height is 8, is %f\r\n",triangleA);
    triangleA = triangleArea(14,5);
    if(triangleA != -1)
        printf ("1. Triangle area, where base is 14 and height is 5, is %f\r\n",triangleA);
    triangleA = triangleArea(-3,7);
    if(triangleA != -1)
        printf ("1. Triangle area, where base is -3 and height is 7, is %f\r\n",triangleA);

    /*TESTING OF CHANGE CASE*/
    printf("\r\nTESTING OF CHANGE CASE\r\n");
    printf ("2. Another case for char 'c' is '%c'\r\n",changeCase('c'));
    printf ("2. Another case for char 'A' is '%c'\r\n",changeCase('A'));
    printf ("2. Another case for char 'f' is '%c'\r\n",changeCase('f'));
    printf ("2. Another case for char 'G' is '%c'\r\n",changeCase('G'));
    printf ("2. Another case for char 'w' is '%c'\r\n",changeCase('w'));
    printf ("2. Another case for char 'X' is '%c'\r\n",changeCase('X'));

    /*TESTING OF LETTER GRADE*/
    printf("\r\nTESTING OF LETTER GRADE\r\n");
    printf ("3. Letter grade for 95 numeric grade is %s\r\n",letterGrade(95));
    printf ("3. Letter grade for 90 numeric grade is %s\r\n",letterGrade(90));
    printf ("3. Letter grade for 65 numeric grade is %s\r\n",letterGrade(65));
    printf ("3. Letter grade for 50 numeric grade is %s\r\n",letterGrade(50));
    printf ("3. Letter grade for 47 numeric grade is %s\r\n",letterGrade(47));
    printf ("3. Letter grade for 150 numeric grade is %s\r\n",letterGrade(150));
    printf ("3. Letter grade for -3 numeric grade is %s\r\n",letterGrade(-3));

    /*TESTING OF HIGHEST POWER DIVISOR*/
    printf("\r\nTESTING OF HIGHEST POWER DIVISOR\r\n");
    int highestPowerDiv = highestPowerDivisor(2,50);
```

```
    if(highestPowerDiv != -1)
        printf ("4. Highest power divisor for 2 and 50 is %i\r\n", highestPowerDiv);
    highestPowerDiv = highestPowerDivisor(5,250);
    if(highestPowerDiv != -1)
        printf ("4. Highest power divisor for 5 and 250 is %i\r\n", highestPowerDiv);
    highestPowerDiv = highestPowerDivisor(25,10);
    if(highestPowerDiv != -1)
        printf ("4. Highest power divisor for 25 and 10 is %i\r\n", highestPowerDiv);
    highestPowerDiv = highestPowerDivisor(7,100);
    if(highestPowerDiv != -1)
        printf ("4. Highest power divisor for 7 and 100 is %i\r\n", highestPowerDiv);
    highestPowerDiv = highestPowerDivisor(-2,13);
    if(highestPowerDiv != -1)
        printf ("4. Highest power divisor for -2 and 13 is %i\r\n", highestPowerDiv);

    /*TESTING OF ADD RANGE*/
    printf("\r\nTESTING OF ADD RANGE\r\n");
    int range=addRange(0, 5, 1);
    if(range != -1)
        printf ("5. Sum for range 0, 5, 1 is %i\r\n",range);
    range=addRange(1, 20, 3);
    if(range != -1)
        printf ("5. Sum for range 1, 20, 3 is %i\r\n",range);
    range=addRange(30, 20, 3);
    if(range != -1)
        printf ("5. Sum for range 30, 20, 3 is %i\r\n",range);
    range=addRange(20, 50, 4);
    if(range != -1)
        printf ("5. Sum for range 20, 50, 4 is %i\r\n",range);
    range=addRange(10, 20, -2);
    if(range != -1)
        printf ("5. Sum for range 10, 20, -2 is %i\r\n",range);

    /*TESTING OF AREA TRIANGLE*/
    printf("\r\nTESTING OF AREA TRIANGLE\r\n");
    double area = areaTriangle(3, 4, 6);
    if(area != -1)
        printf ("6. Triangle area for sides a=3, b=4 and c=6 is %f\r\n",area);
    area = areaTriangle(3, 4, 2);
    if(area != -1)
        printf ("6. Triangle area for sides a=3, b=4 and c=2 is %f\r\n",area);
    area = areaTriangle(2, 7, 8);
    if(area != -1)
        printf ("6. Triangle area for sides a=2, b=7 and c=8 is %f\r\n",area);
    area = areaTriangle(14, 15, 12);
    if(area != -1)
        printf ("6. Triangle area for sides a=14, b=15 and c=12 is %f\r\n",area);
    area = areaTriangle(7, 15, 7);
    if(area != -1)
        printf ("6. Triangle area for sides a=7, b=15 and c=7 is %f\r\n",area);
    return 0;
}
/*function 'triangleArea' count area of triangle clasiclly s= (a*v)/2
* input parameters are floats which are >0 (there is a control if parameters are > 0)
* and return value has also type Float*/
float triangleArea(float a, float v)
{
```

```
    if(a <= 0 || v <= 0)
    {
        perror("1. Base and height have to be greater than 0");
        return -1;
    }
    return (a*v)/2;
}
/*function 'changeCase' change the letter case
* as an input char is expected and according to his case will be this letter changed to another case
* if is in lower case, than is changed to upper case,
* otherwise is expected as char in upper case and changed to lower case*/
char changeCase(char a)
{
    if(islower(a))
        return toupper(a);
    else
        return tolower(a);
}
/*function 'letterGrade' expected as input integer from 0 to 100
* returned value is String according to input number,
* if is lower than 0 or greater than 100, an error is thrown*/
char * letterGrade(int grade)
{
    if(grade >=90 && grade <=100)
        return "A";
    else if(grade >=80 && grade <=89)
        return "B";
    else if(grade >=70 && grade <=79)
        return "C";
    else if(grade >=60 && grade <=69)
        return "D";
    else if(grade >= 0 && grade <=59)
        return "E / F";
    else
    {
        perror("3. Bad numeric grade");
        return "does not exist";
    }
}
/*function 'highestPowerDivisor' expected as input two integers > 0
* (there is a control if parameters are > 0)
* are expected and returned value is also integer
* if first number modulo second number is nonzero, then the power is zero
* otherwise is first number modulo second number is zero, the power is at least 1
* so the result is 1 + and recursively count the power
* for first number and for second number divided by first number*/
int highestPowerDivisor(int number1, int number2)
{
    if(number1 <= 0 || number2 <= 0)
    {
        perror("4. Input parameters must > 0");
        return -1;
    }
    if(number2 % number1 != 0)
        return 0;
    else
        return 1 + highestPowerDivisor(number1, number2/number1);
```

```
}
/*function 'addRange' expected as input input 3 integers are expected
* return value is also integer
* if first argument is greater than second argument
* or third argument lower than zero, an error is thrown
* if first argument plus third argument are higher than second argument,
* function returns first argument because second argument is border of range
* else if first argument plus third argument are equal to second argument,
* function returns first argument plus second argument
* (it is also possible return twice first argument plus third argument)
* else if first plus third argument is lower than second argument,
* we add first argument to recursion call where first argument is previous first
* argument * plus added range ( = third argument ), other parameters are same*/
int addRange(int a, int b, int c)
{
      int error= 0;
      if(a > b)
      {
      perror("5. First parameter must be  than second parameter");
      error=1;
      }
      if(c < 0)
      {
          perror("5. Third parameter must be > than 0");
      error=1;
      }
      if(error)
          return -1;
      if(a + c > b)
        return a;
      else if(a + c == b)
        return a + b;
      else if(a + c < b)
        return a + addRange((a + c), b, c);
}
/*function 'areaTriangle' expects three Floats
* which are length of concrete side of triangle and returns value which is also float
* firstly it is tested if values of sides describe a trinagle
* if yes, area of triangle is counted using parameter 's' counted before*/
double areaTriangle(double a, double b, double c)
{
    if((a + b)<= c || (a + c)<=b || (b + c)<=a)
    {
        perror("6. Sides a, b, c do not describe a triangle");
        return -1;
    }
    double s= (a+b+c)/2;
    return sqrt(s*(s-a)*(s-b)*(s-c));
}
```