Assignment\_02\_Shoyeb\_Khan\_2021H1540806P

Shoyeb\_Khan\_2021H1540806P

02/10/2021

#1. Write a function that takes two numbers, a and b, as arguments, and returns a raised to the power b.  
  
b\_pow\_a <- function(a,b)  
 {return (b\*\*a) ; }  
  
#+ Challenge 1: Try to generate the numbers a and b randomly in the range of 1 to 10 as part of the function.  
a <- sample(1:10,10)  
  
b <- sample(1:10,10)  
  
a # The value of a.

## [1] 1 5 7 9 6 10 3 8 4 2

b # The value of b.

## [1] 6 4 5 3 9 10 8 7 1 2

#+ Challenge 2: Initialize the number a as a vector and try to return a vector with each element of a raised to the number b.  
b\_pow\_a(a,b)

## [1] 6 1024 78125 19683 531441 10000000000  
## [7] 512 5764801 1 4

#2. Write a function that takes a number as argument and returns true if it is divisible by 2, 3, 5, 7, 11, 13, and false otherwise.  
   
# + Challenge: Randomize the number that needs to be given as an argument to the function in the range from 1 to 1000.  
  
  
divisiblity\_check <- function(x)  
{  
 if(x %%2 || x%%3 || x%%5 || x%%7 || x%%11 || x%%13)  
 {  
 check\_o <- "TRUE"  
 }  
   
 else   
 check\_o <- "FALSE"  
   
 return (check\_o) ;  
   
}  
  
  
rand\_num <- sample(1:1000,1000)  
checker <- c()  
  
for ( i in 1:1000){  
checker[i] <- divisiblity\_check(rand\_num[i])  
   
}  
  
#3. Write a function that takes a non-negative integer as its argument and returns its factorial.  
# + Challenge: Do not use the built-in function factorial()  
  
factorial\_num <- function(y)  
{  
 if ( y ==0 || y==1 )  
   
 return(1)  
   
 else   
 return (y\*factorial\_num(y-1))  
   
   
 }  
  
#4. Write a function to compute the dot product of two vectors.  
#+ Challenge 1: Do not use in-built R functions.(Clue: Iterative algorithm)  
  
dot\_prod\_func <- function(a,b)  
{  
 prod <- c()  
  
 for (i in 1 :3) {  
 prod[i] <- a[i] \* b[i]  
 }  
  
 sum <- 0  
  
 for (j in 1 :3) {  
 sum <- sum + prod[i]  
 }  
  
 return(sum)  
   
}  
  
  
a <- c(1,2,3)  
b <- c(4,5,9)  
  
dot\_prod\_func(a,b)

## [1] 81

#+ Challenge 2: Vectors should be of arbitrary length.  
  
##Enter any arbitary number and store it in z\_vector.  
  
#z <- scan()  
#z\_p <- scan()  
  
z <- c(1,2,3,4,5)  
  
z\_p <- c(9,8,7,6,5)  
  
dot\_prod\_arbi <- function(z,z\_p){  
 prod\_arbi <- c()  
   
 for(i in 1 : length(z)){  
 prod\_arbi[i] <- z[i]\*z\_p[i]  
 }  
  
 sum\_arbi <- 0  
  
 for (j in 1 :length(z)) {  
 sum\_arbi <- sum\_arbi + prod\_arbi[j]  
 }  
   
 return(sum\_arbi)  
   
}  
  
sprintf("The dot product of two vectors z and z\_p is : %d",dot\_prod\_arbi(z,z\_p))

## [1] "The dot product of two vectors z and z\_p is : 95"

#5. Write a function that takes a numeric vector and prints on screen the mean and standard deviation.  
#+ Challenge: Do not use in-built R functions.  
#n <- readline()  
#n <- as.integer(n)  
  
n <- 10  
  
data <- sample(1:n,n)  
  
mean\_sd\_value <- function(data,n){  
 sum\_data <- 0  
  
 for(i in 1:n){  
 sum\_data <- sum\_data + data[i]  
 }  
  
 mean <- sum\_data/10  
  
 sprintf("The mean of the sample data is :%f",mean)  
   
 ## Calculation of the Standard Deviations.  
  
 diff\_squ <- c()  
 sum\_diff <- c()  
  
 sum\_diff <- 0  
  
 for(i in 1:n){  
 diff\_squ[i] <- (data[i] - mean)\*\*2   
 sum\_diff <- sum\_diff + diff\_squ[i]  
 }  
  
 variance <- sum\_diff/(n-1)  
  
 standard\_dev <- sqrt(variance)  
  
 sprintf("The standard deviation of the sample data is :%f",standard\_dev)  
   
}  
  
mean\_sd\_value(data,n)

## [1] "The standard deviation of the sample data is :3.027650"

#6. Write a function given that takes a vector as its argument and returns a new vector with the elements of the first vector with #duplicated elements removed.   
# + HINT: Iterate through the vector checking for duplicate elements.  
  
vector\_uni <- function(vector\_dup)  
{  
 vector\_undup <- c()  
  
 for(i in 1:length(vector\_dup))  
 {  
 if(sum(vector\_dup[i]==vector\_undup) == 0)  
 {   
 vector\_undup <- c(vector\_undup,vector\_dup[i])  
   
 }  
 }  
 return (vector\_undup)  
   
}  
  
vector\_dup <- c(1,12,12,15,16,9,9,10,20,100,100,56)  
  
vector\_uni(vector\_dup)

## [1] 1 12 15 16 9 10 20 100 56

#7. Write a function to find out the factors of a positive integer.   
#+ Challenge: Find 2 different solutions.   
#+ HINT (For 1st solution): Look up the R documentation for functions such as `lapply`  
  
factors\_fun <- function(x){  
   
 fact <- c()  
 fact <- 0  
   
 for(i in 1:x){  
 if(x %% i == 0){  
 fact[i] <- i   
 }  
 }  
   
 return (na.omit(fact))  
   
}  
  
list <- c(1,5,8,10,12)  
  
lapply(list, factors\_fun)

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] 1 5  
## attr(,"na.action")  
## [1] 2 3 4  
## attr(,"class")  
## [1] "omit"  
##   
## [[3]]  
## [1] 1 2 4 8  
## attr(,"na.action")  
## [1] 3 5 6 7  
## attr(,"class")  
## [1] "omit"  
##   
## [[4]]  
## [1] 1 2 5 10  
## attr(,"na.action")  
## [1] 3 4 6 7 8 9  
## attr(,"class")  
## [1] "omit"  
##   
## [[5]]  
## [1] 1 2 3 4 6 12  
## attr(,"na.action")  
## [1] 5 7 8 9 10 11  
## attr(,"class")  
## [1] "omit"

## 2 HINT (For 2nd solution): `Vectorize` and `Filter` functions may be of some help.  
  
vector\_fun <- Vectorize(factors\_fun)  
  
vector\_fun(3)

## [,1]  
## [1,] 1  
## [2,] 3

#8. Write a function that prints out the value of the Ackermann Function after entering two numbers as input. (Picked up this problem because it # #reminds me of 'Attack on Titan', PS: It's an anime!)   
  
ackerman\_func <- function(m,n){  
 if(m == 0){  
 n +1   
 }  
   
 else if(m >0 && n ==0){  
 ackerman\_func(m-1,1)  
 }  
   
 else if(m > 0 && n > 0){  
 ackerman\_func(m-1,ackerman\_func(m,n-1))  
 }  
   
}  
ackerman\_func(1,200)

## [1] 202

ackerman\_func(1,10)

## [1] 12

ackerman\_func(1,7)

## [1] 9