Ericka Najera

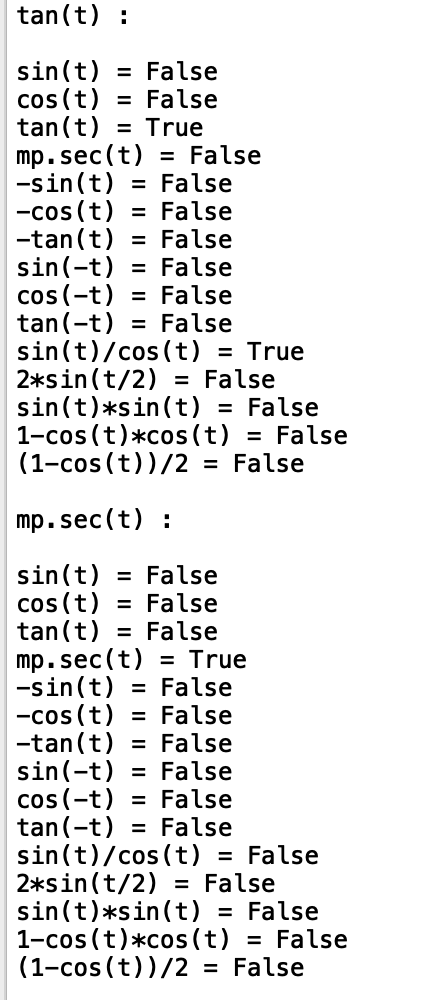
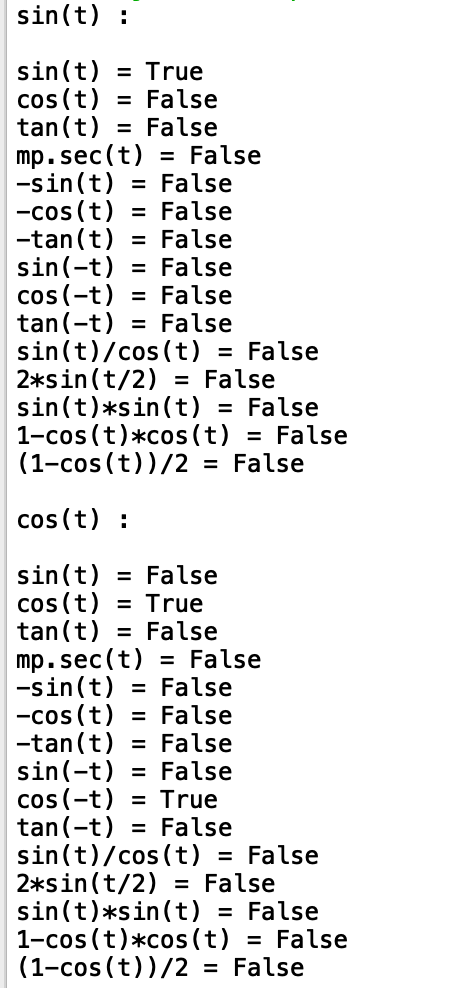
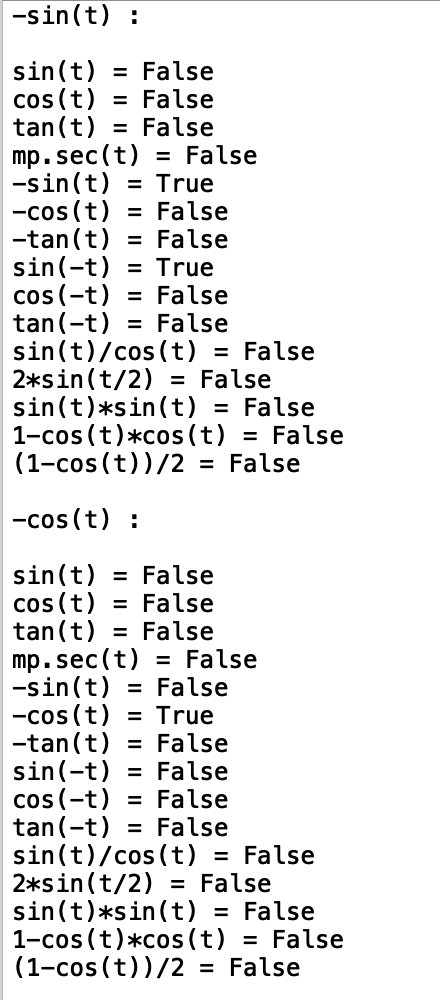
CS 2302

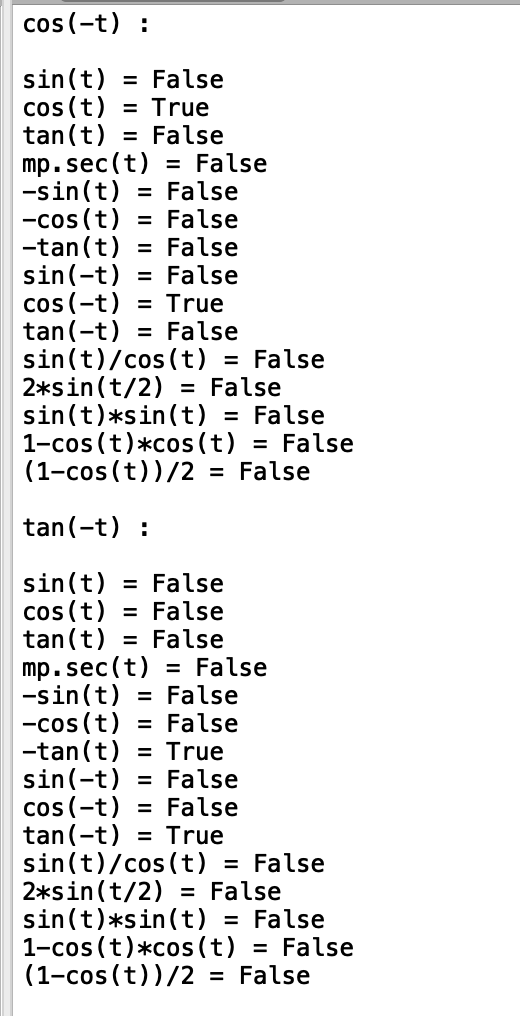
14 May 2019

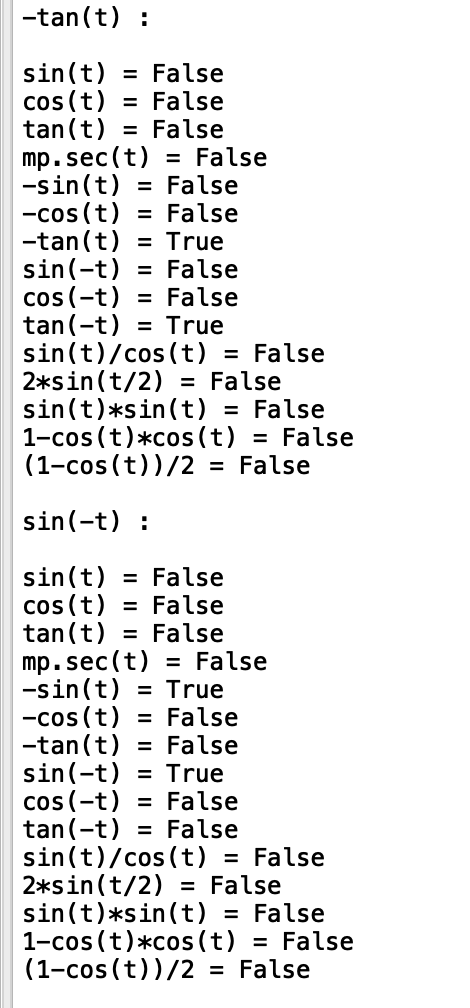
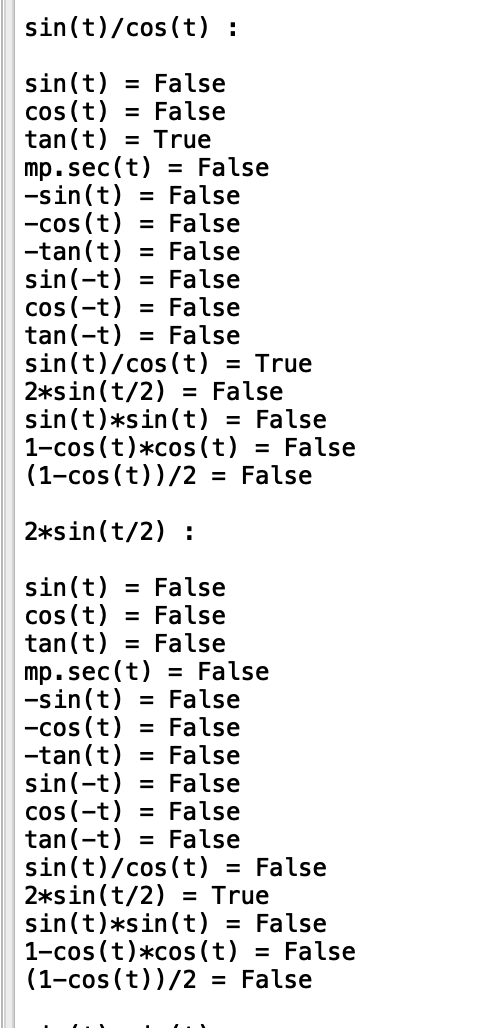
Lab 8

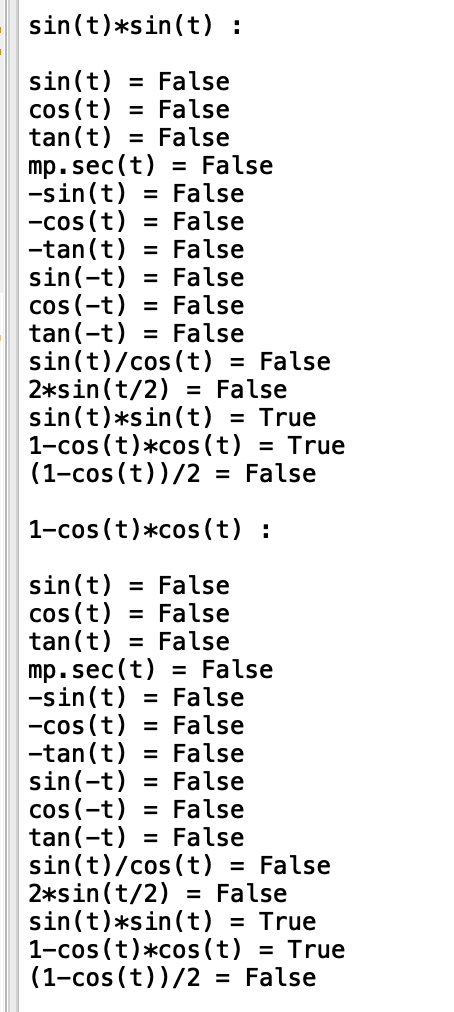
Lab 8 was an implementation of functions using backtracking and randomized programming. The first function is to get a list of functions and check them with each other and determine if they are identities or not. The seconds function was to receive a set as a parameter and use backtracking to see if you can split it in two sets and have the two sets with the same sum. If no equal sum is possible then we return an empty array.

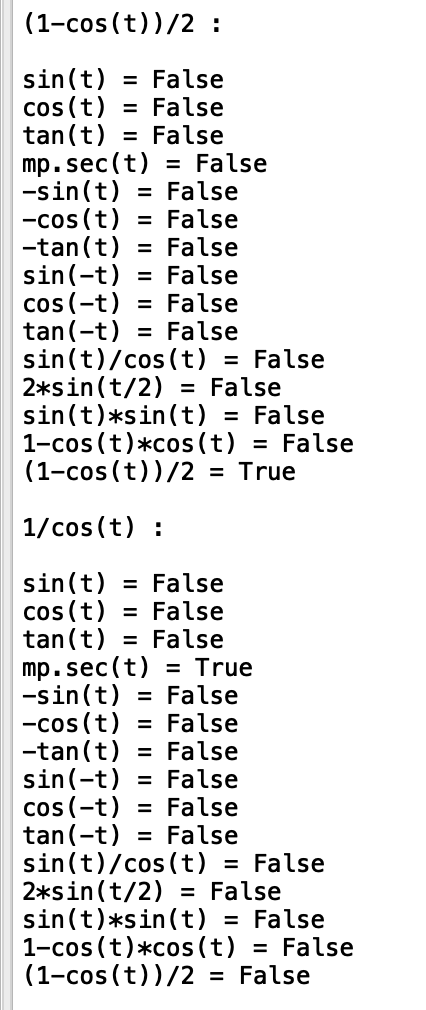
For the method with randomization to find the identities at first I placed the functions in the list in order to compare them. I order to traverse and compare them, I though about placing two loops in order to traverse for each function and then compare each function with the rest the list. At first I created a while loop with the counter in order to increment the counter when the function would check itself but I couldn’t get it to skip appropriately so I decided to just let it check itself for each function. When we check if it is an identity I save the two functions and get a random number from -pi to pi. Then I evaluate them and check the difference. I do that around 1000 times to check if they are really equal. After I check I append the function along with true or false, to save the result of it being an identity or not. After I return the results I made another function to display the results. I did two for loops to traverse the list and display the answer. I displayed the function and the other combinations and it was true or false.



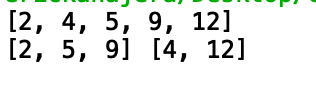


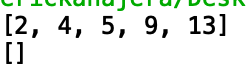






The function where I need to separate a set into subsets with equal sums. I tried a few ways but somehow it wouldn’t give me the right result. At first I needed to check the sum of the set. If it was an even number then it would give me split, and if it wasn’t them there would be no equal sum in the subsets therefore it would have to return an empty set. If it could spliy then I would sent to another method that split it. At first I tried append to one subset and check the sum, and it it wouldn’t give the correct sum, then it would pop and recursively backtrack to find the answer, but somehow my second set would have repeated numbers and not the correct output. I realized that the subset sum method would return one subset with a specific goal. Then I realized that I could send the set and half the sum as my goal. From there I would return the set and from there I made another function to pop the items from the set that were already in the subset and the remaining numbers of the set would be subset number 2. After that I would just print the result of the two subsets and it if it was not possible an empty set would be printed.





Running times:

Identities: O(n^2)

Print results: O(n^2)

Partition : O(n^3 )

Appendix:

﻿ # Ericka Najera Lab 8 MW 10:30-11:50

#Professor Fuentes CS2302

#Lab 8 implement backtracking and randomized programming

import random

import numpy as np

from math import \*

from mpmath import \*

#this method returns a list of lists that compares all the functions with each other

def Identities(F):

#create a list of lists to store rewsults

results =[ [] for i in range(len(F)) ]

#create a for loop to traverse the list of functions

for i in range(len(F)):

#create a while loop to check each function with each other

counter =0

#append the function we are checking

results[i].append(F[i])

f1 = F[i]

while counter < len(F):

#check both functions by evaluating with a random number from -pi to pi

f2 = F[counter]

similar = True

for n in range(1000):

t = random.randrange(int(-math.pi),int(math.pi))

y1 = eval(f1)

y2 = eval(f2)

if np.abs(y1-y2)>0.0001:

similar = False

counter+=1

#append if it is true or false

results[i].append([f2,similar])

return results

def similarties(L):

#receive the list and print the results accordingly

for i in range(len(L)):

print(L[i][0],':')

print()

for j in range(1,len(L)):

print(L[i][j][0],'=',L[i][j][1])

print()

#return a subset with the acordin sum

def subsetsum(S,last,goal):

#base case

if goal ==0:

return True, []

if goal<0 or last<0:

return False, []

#save result into a variable and traverse the set, goal gets less my the number appended

res, subset = subsetsum(S,last-1,goal-S[last])

if res:

subset.append(S[last])

return True, subset

else:

return subsetsum(S,last-1,goal)

#creates the two sets from the set already created and the initial set

def split(S,set1):

#remove the subset from the set

for i in range(len(set1)):

if set1[i] in S:

S.remove(set1[i])

#return the new two subsets

print(S,set1)

def Partition(S, n) :

#gets the sum of the array

sum = 0

#adds each element of the array into the sum

for i in range(n):

sum += S[i]

#if the sum is not even then no partition can be made

if (sum % 2 != 0) :

return False

#sends the set to be divided into a subset with half the sum

return subsetsum(S,n-1,sum//2)

S = [2, 4, 5, 9,13]

print(S)

if Partition(S,len(S))== False:

print('[]')

else:

split(S,set1)

F = ['sin(t)','cos(t)','tan(t)','mp.sec(t)','-sin(t)','-cos(t)','-tan(t)','sin(-t)','cos(-t)','tan(-t)','sin(t)/cos(t)','2\*sin(t/2)','sin(t)\*sin(t)','1-cos(t)\*cos(t)','(1-cos(t))/2','1/cos(t)']

sim = Identities(F)

similarties(sim)

I Ericka Najera certify that this project is entirely my own work. I wrote, debugged, and teste the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

