LAB 8

CS2302 – Data Structures

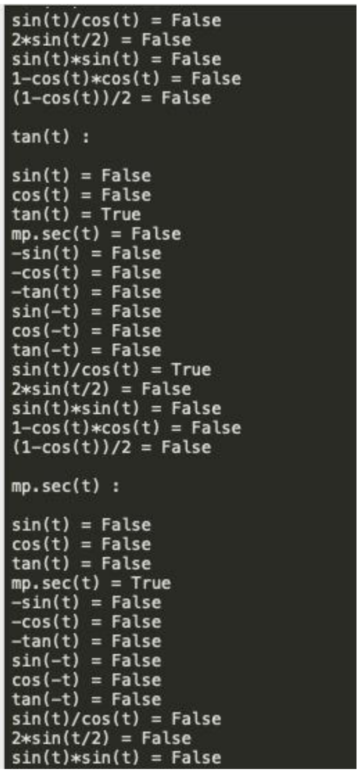
Quinones Rodriguez, Diego A

Report

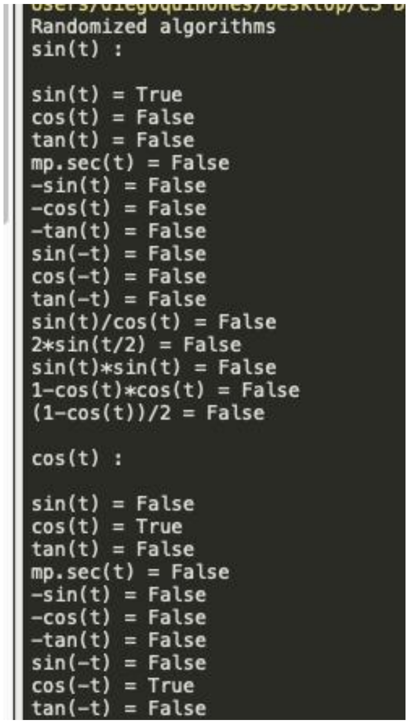
Lab 8 consisted of using 2 of the recently seen programming methods, the first one being randomized algorithms. We had to use multiple trigonometric identities and test them; we tested them with random numbers, and random algorithms.

The second part we had to check if a set of numbers and check if we could divide it in two sets. But the relevant part is that we had to use backtracking. Based on if the partition was possible we would display it.

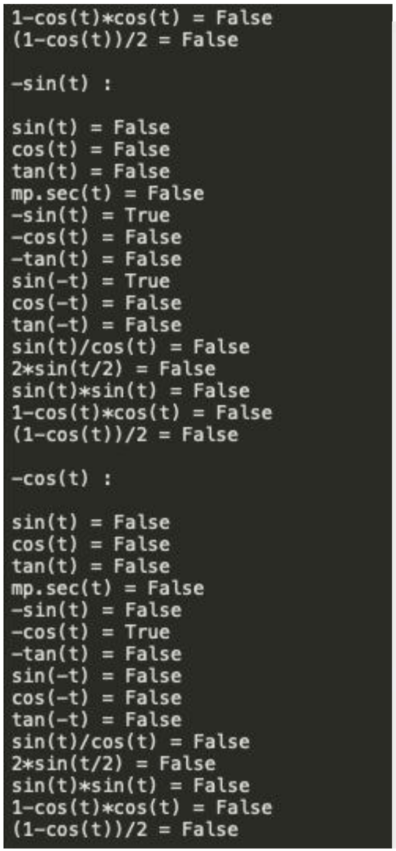
Screenshots



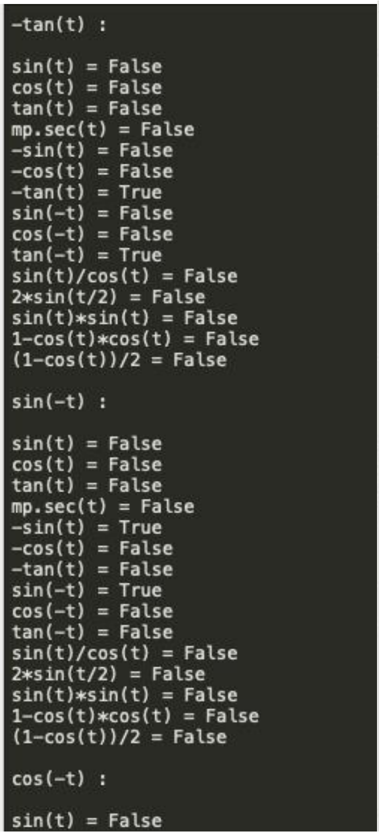
```
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

tan(t) :

sin(t) = False
cos(t) = False
tan(t) = True
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = True
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

mp.sec(t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = True
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
```



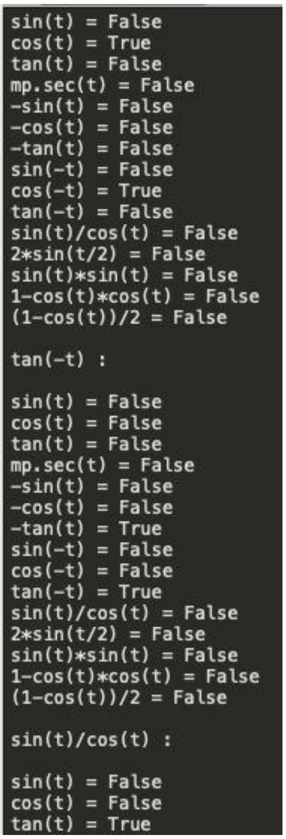
```
Randomized algorithms
sin(t) :

sin(t) = True
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

cos(t) :

sin(t) = False
cos(t) = True
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = True
tan(-t) = False
```



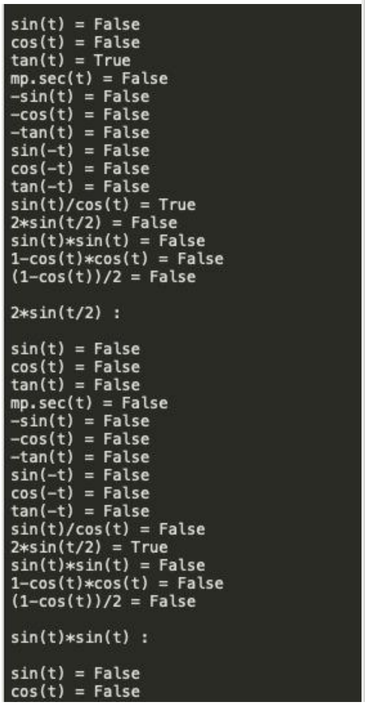
```
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

-sin(t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = True
-cos(t) = False
-tan(t) = False
sin(-t) = True
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

-cos(t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = True
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False
```



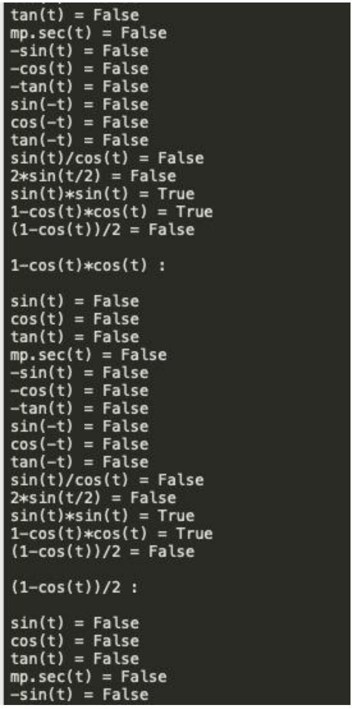
```
-tan(t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = True
sin(-t) = False
cos(-t) = False
tan(-t) = True
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

sin(-t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = True
-cos(t) = False
-tan(t) = False
sin(-t) = True
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

cos(-t) :

sin(t) = False
```



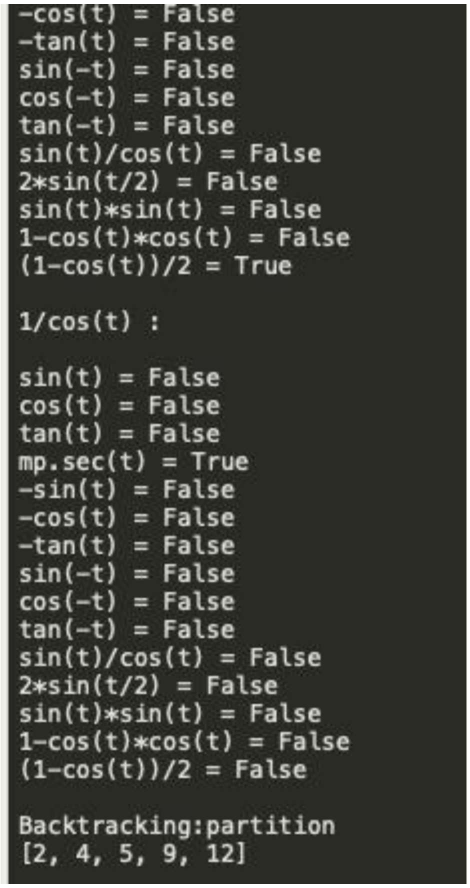
```
sin(t) = False
cos(t) = True
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = True
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

tan(-t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = True
sin(-t) = False
cos(-t) = False
tan(-t) = True
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

sin(t)/cos(t) :

sin(t) = False
cos(t) = False
tan(t) = True
```



```
sin(t) = False
cos(t) = False
tan(t) = True
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = True
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

2*sin(t/2) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = True
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

sin(t)*sin(t) :

sin(t) = False
cos(t) = False
```



```
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = True
1-cos(t)*cos(t) = True
(1-cos(t))/2 = False

1-cos(t)*cos(t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = True
1-cos(t)*cos(t) = True
(1-cos(t))/2 = False

(1-cos(t))/2 :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = False
-sin(t) = False
```



```
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = True

1/cos(t) :

sin(t) = False
cos(t) = False
tan(t) = False
mp.sec(t) = True
-sin(t) = False
-cos(t) = False
-tan(t) = False
sin(-t) = False
cos(-t) = False
tan(-t) = False
sin(t)/cos(t) = False
2*sin(t/2) = False
sin(t)*sin(t) = False
1-cos(t)*cos(t) = False
(1-cos(t))/2 = False

Backtracking:partition
[2, 4, 5, 9, 12]
```

Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May 13 13:33:36 2019
@author: diegoquinones
"""
#Lab



import random
import numpy as np
from math import *
from mpmath import *
```

```
def Identities(F):
    #this is where answers would be stored
    results =[ [] for i in range(len(F)) ]
    for i in range(len(F)):
        counter =0
        results[i].append(F[i])
        f1 = F[i]
        while counter < len(F):
            f2 = F[counter]
            similar = True
            for n in range(1000):
                #creates random value
                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
            results[i].append([f2,similar])
            #stores true/false
    return results


def similarties(L):
    #prints the trig function
    for i in range(len(L)):
        print(L[i][0],':')
        print()
        #prints similarities
        for j in range(1,len(L)):
            print(L[i][j][0],'=',L[i][j][1])
        print()


def sums(S,last,goal):
    if goal ==0:
        #we want to get to 0 so it return true
        return True, []
    if goal<0 or last<0:
        return False, []
    res, subset = sums(S,last-1,goal-S[last])
    if res:
        subset.append(S[last])
        return True, subset
    else:
        #recursive call
        return sums(S,last-1,goal)

def partition(S, n) :
    s = 0
    for i in range(n):
        s += S[i]
    if (s % 2 != 0) :
        return False
    return sums(S,n-1,s//2)

def split(S,set1):
    for i in range(len(set1)):
        #if the value is already in S it gets removed
        if set1[i] in S:
            S.remove(set1[i])
    print(S,set1)


print('Randomized algorithms')


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)


print('Backtracking:partition')


S = [2,4,5,9,12]
print(S)


if partition(S,len(S))== False:
    print('[]')
```

Academic Honesty Certification

“I certify that this project is entirely my own work. I wrote, debugged, and tested 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.” -Diego Quinones