**CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF TECHNOLOGY & ENGINEERING**

**DEVANG PATEL INSTITUTE OF ADVANCED TECHNOLOGY AND RESEARCH**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SUBJECT NAME:** ARTIFICIAL INTELLIGENCE

**SUBJECT CODE:** CS341

**SEMESTER:** V

**ACADEMIC YEAR:** 2020-21

**PRACTICAL - 1**

**AIM : Write Programs to demonstrate knowledge of Python Basics.**

* 1. **Write a program to demonstrate variable creation in python**

**PROGRAM CODE**

**#assigning an integer**

**id = 7**

**#assignig string**

**name = "Rudra Barad"**

**#assigning an floating point**

**decimal = 80008.70101**

**print(id)**

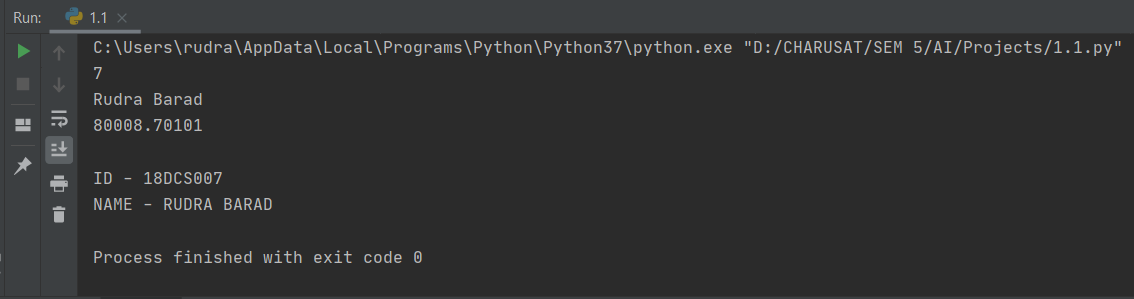
**print(name)**

**print(decimal)**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD")**

**OUTPUT**



* 1. **Write a program to demonstrate command input in python**

**PROGRAM CODE**

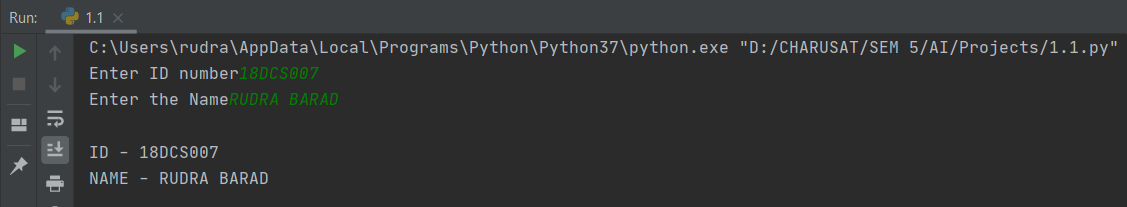
**id = input("Enter ID number");**

**name = input("Enter the Name")**

**print("\nID - "+id)**

**print("NAME - "+name)**

**OUTPUT**



* 1. Write a program to demonstrate numbers and stings in python

**PROGRAM CODE**

**# int**

**a = 123**

**# float**

**b = 456.789**

**print(a)**

**print(b)**

**str = 'Rudra'**

**print(str)**

**str = "Barad"**

**print(str)**

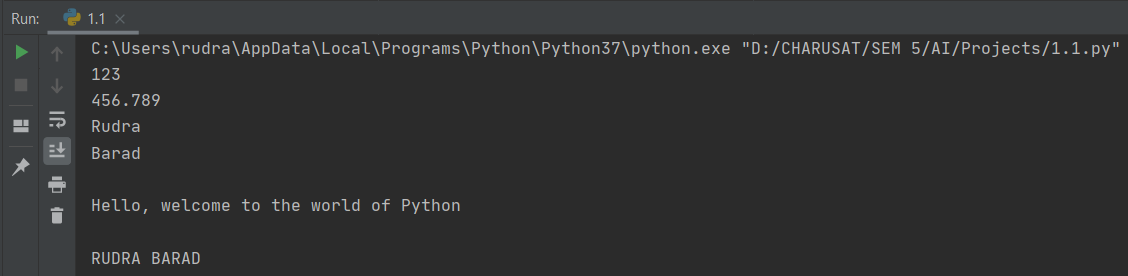
**string = """\nHello, welcome to the world of Python"""**

**print(string)**

**str = '''\nRUDRA BARAD'''**

**print(str)**

**OUTPUT**



* 1. Write a program to demonstrate operators in python.

**PROGRAM CODE**

**a = 10**

**b = 7**

**c = 5**

**d = 5**

**e = 1**

**add = a+b**

**print(add)**

**sub = c-d**

**print(sub)**

**mul = e\*a**

**print(mul)**

**div1 = b/c**

**print(div1)**

**div2 = b//c**

**print(div2)**

**mod = d%e**

**print(add)**

**# Examples of Relational Operators**

**x = 13**

**y = 33**

**# x > y is False**

**print(x > y)**

**# x < y is True**

**print(x < y)**

**# x == y is False**

**print(x == y)**

**# x != b is True**

**print(x != y)**

**# x >= y is False**

**print(x >= y)**

**# x <= y is True**

**print(x <= y)**

**# Examples of Logical Operator**

**p = True**

**q = False**

**# Print p and q is False**

**print(p and q)**

**# Print p or q is True**

**print(p or q)**

**# Print not p is False**

**print(not p)**

**# Examples of Bitwise operators**

**r = 10**

**s = 4**

**# Print bitwise AND operation**

**print(r & s)**

**# Print bitwise OR operation**

**print(r | s)**

**# Print bitwise NOT operation**

**print(~r)**

**# print bitwise XOR operation**

**print(r ^ s)**

**# print bitwise right shift operation**

**print(r >> 2)**

**# print bitwise left shift operation**

**print(r << 2)**

**# Examples of Identity operators**

**a1 = 3**

**b1 = 3**

**a2 = 'charusatuniversity'**

**b2 = 'charusatuniversity'**

**a3 = [1, 2, 3]**

**b3 = [1, 2, 3]**

**print(a1 is not b1)**

**print(a2 is b2)**

**# Output is False, since lists are mutable.**

**print(a3 is b3)**

**# Examples of Membership operator**

**x1 = 'charusatuniversity'**

**y1 = {3: 'c', 4: 's'}**

**print('s' in x1)**

**print('charusat' not in x1)**

**print('charusat' not in x1)**

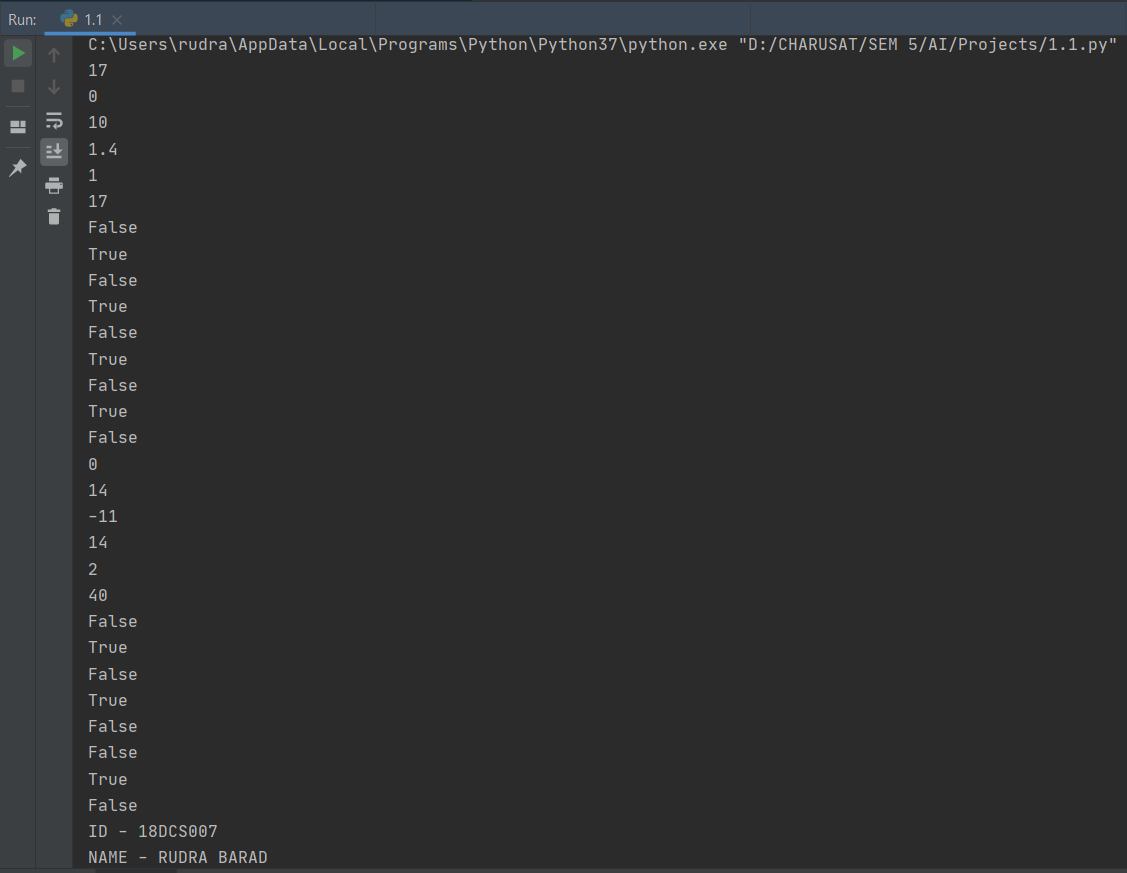
**print(3 in y1)**

**print('c' in y1)**

**print("ID - 18DCS007")**

**print("NAME - RUDRA BARAD")**

**OUTPUT**



* 1. Write a program to demonstrate decision making in python

**PROGRAM CODE**

**i = 10**

**if (i > 15):**

**print ("10 is less than 15")**

**#If else statement**

**j = 20;**

**if (j < 15):**

**print ("J(20) is smaller than 15")**

**else:**

**print ("J(20) is greater than 15")**

**# nested If statement**

**k = 10**

**if (k == 10):**

**if (k < 15):**

**print ("K(10) is smaller than 15")**

**if (k < 12):**

**print ("K(10) is smaller than 12")**

**else:**

**print ("K(10) is greater than 15")**

**#if-elif-else ladder**

**m = 20**

**if (m == 10):**

**print ("M(20) is 10")**

**elif (m == 15):**

**print ("M(20) is 15")**

**elif (m == 20):**

**print ("M(20) is 20")**

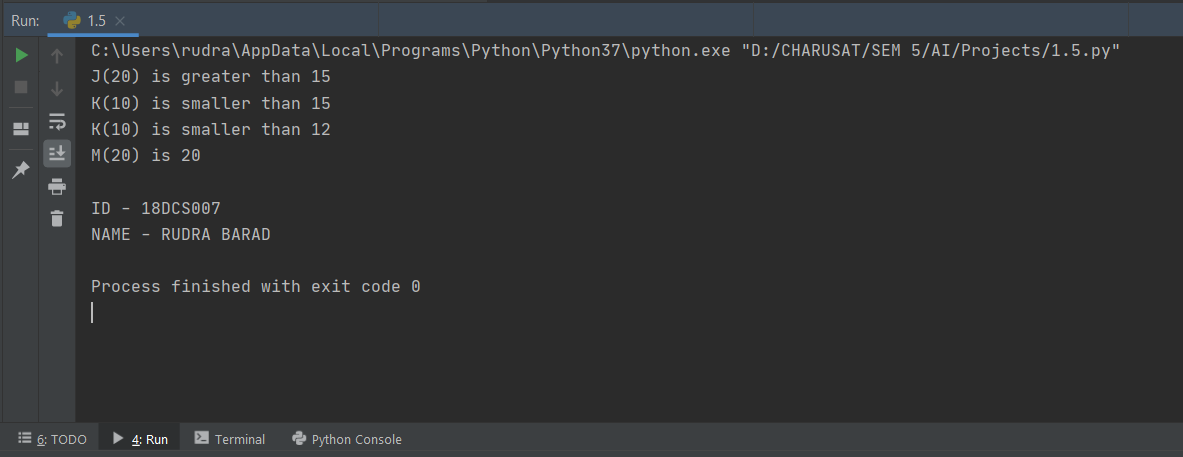
**else:**

**print ("M(20) is not present")**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD")**

**OUTPUT**



* 1. Write a program to demonstrate control structures in python

**PROGRAM CODE**

**# while loop**

**count = 0**

**while (count < 3):**

**count = count + 1**

**print("in loop")**

**l = ["Rudra", "Barad"]**

**for i in l:**

**print(i)**

**# Iterating over a tuple (immutable)**

**arr = ("Curio", "Rimor")**

**for i in arr:**

**print(i)**

**# Iterating over a String**

**str = "RUDRA"**

**for i in str:**

**print(i)**

**# Iterating over dictionary**

**d = dict()**

**d['xyz'] = 123**

**d['abc'] = 345**

**for i in d:**

**print("%s %d" % (i, d[i]))**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD")**

**OUTPUT**



* 1. Write a program to demonstrate lists and dictionary in python

**PROGRAM CODE**

**li = [7, "Rudra", 7+7]**

**print (li)**

**li.append(28)**

**print (li)**

**li.pop()**

**print (li)**

**thisdict = {**

**"class": "5CSE1",**

**"name": "Rudra",**

**"year": 2020**

**}**

**x = thisdict["class"]**

**print("\n"+x+"\n")**

**thisdict["year"] = 2021**

**print(thisdict)**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD")**

**OUTPUT**



**PRACTICAL - 2**

**AIM : Write a program to solve Tower of Hanoi problem in python**

**PROGRAM CODE**

**def hanoi(disks, source, auxiliary, target):**

**if disks == 1:**

**print('Move disk 1 from peg {} to peg {}.'.format(source, target))**

**return**

**hanoi(disks - 1, source, target, auxiliary)**

**print('Move disk {} from peg {} to peg {}.'.format(disks, source, target))**

**hanoi(disks - 1, auxiliary, source, target)**

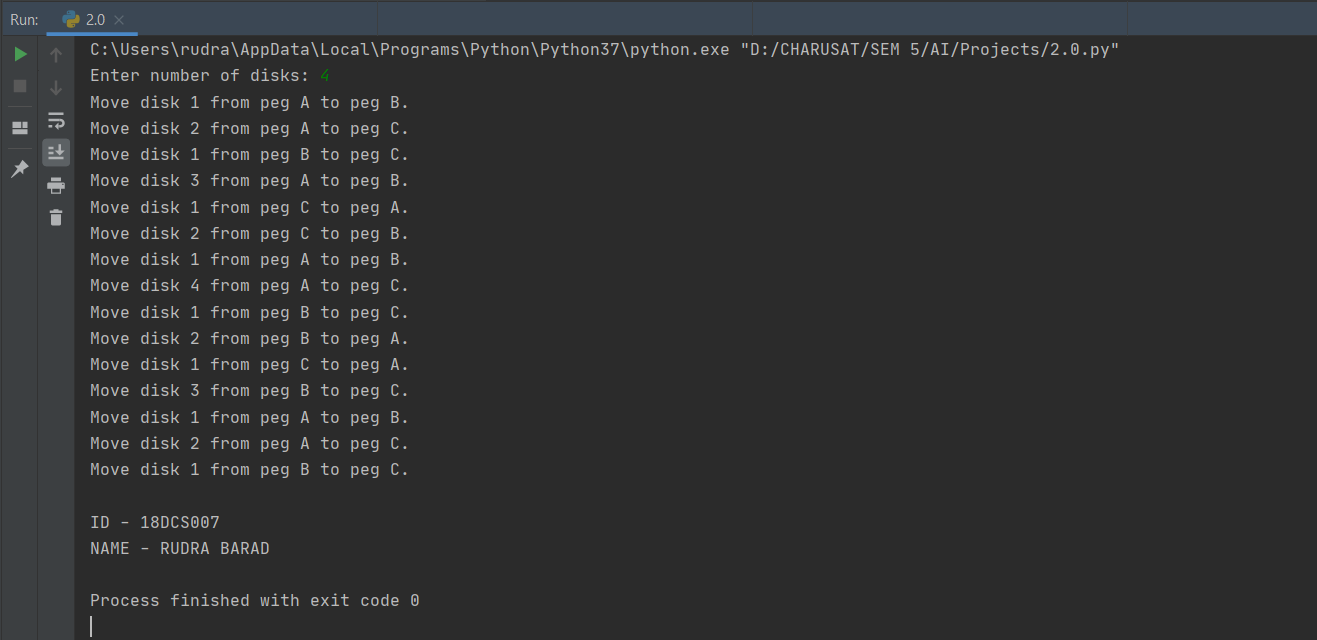
**disks = int(input('Enter number of disks: '))**

**hanoi(disks, 'A', 'B', 'C')**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD")**

**OUTPUT**



**PRACTICAL – 3**

**3.1 AIM : Design Depth First Search Tree and Breadth First Search Tree for Water-Jug Problem in python**

**BFS APPROACH**

**PROGRAM CODE**

**print ( "\nWATER JUG PROBLEM \n" )**

**x\_capacity = int ( input ( "Enter Jug 1 capacity: " ))**

**y\_capacity = int ( input ( "Enter Jug 2 capacity: " ))**

**end = int ( input ( "Enter target volume: " ))**

**def bfs(start, end, x\_capacity, y\_capacity):**

**path = []**

**front = []**

**front.append(start)**

**visited = []**

**#visited.append(start)**

**while ( not ( not front)):**

**current = front.pop()**

**x = current[ 0 ]**

**y = current[ 1 ]**

**path.append(current)**

**if x == end or y == end:**

**print ( "\nFOUND ! \n" )**

**return path**

**# rule 1**

**if current[ 0 ] < x\_capacity and ([x\_capacity, current[ 1 ]] not in visited):**

**front.append([x\_capacity, current[ 1 ]])**

**visited.append([x\_capacity, current[ 1 ]])**

**# rule 2**

**if current[ 1 ] < y\_capacity and ([current[ 0 ], y\_capacity] not in visited):**

**front.append([current[ 0 ], y\_capacity])**

**visited.append([current[ 0 ], y\_capacity])**

**# rule 3**

**if current[ 0 ] > x\_capacity and ([ 0 , current[ 1 ]] not in visited):**

**front.append([ 0 , current[ 1 ]])**

**visited.append([ 0 , current[ 1 ]])**

**# rule 4**

**if current[ 1 ] > y\_capacity and ([x\_capacity, 0 ] not in visited):**

**front.append([x\_capacity, 0 ])**

**visited.append([x\_capacity, 0 ])**

**# rule 5**

**#(x, y) -> (min(x + y, x\_capacity), max(0, x + y - x\_capacity))**

**if current[ 1 ] > 0 and ([ min (x + y, x\_capacity), max ( 0 , x + y - x\_capacity)] not in visited):**

**front.append([ min (x + y, x\_capacity), max ( 0 , x + y - x\_capacity)])**

**visited.append([ min (x + y, x\_capacity), max ( 0 , x + y - x\_capacity)])**

**if current[ 0 ] > 0 and ([ max ( 0 , x + y - y\_capacity), min (x + y, y\_capacity)] not in visited):**

**front.append([ max ( 0 , x + y - y\_capacity), min (x + y,y\_capacity)])**

**visited.append([ max ( 0 , x + y - y\_capacity), min (x + y,y\_capacity)])**

**return "Not found"**

**def gcd(a,b):**

**if a==0:**

**return b**

**return gcd(b%a,a)**

**start = [ 0 , 0 ]**

**if end % gcd(x\_capacity,y\_capacity) == 0 :**

**print (bfs(start, end, x\_capacity, y\_capacity))**

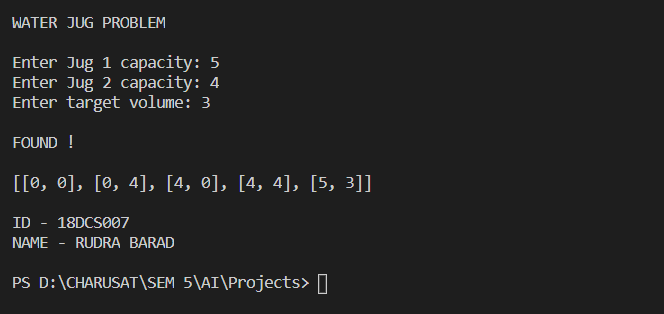
**else :**

**print ( " No solution possible for this combination. " )**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD\n")**

**OUTPUT**



**DFS APPROACH**

**PROGRAM CODE**

**capacity = (10, 9, 8)**

**x = capacity[0]**

**y = capacity[1]**

**z = capacity[2]**

**memory = {}**

**ans = []**

**def get\_all\_states(state):**

**a = state[0]**

**b = state[1]**

**c = state[2]**

**if a == 6 and b == 6:**

**ans.append(state)**

**return True**

**if (a, b, c) in memory:**

**return False**

**memory[(a, b, c)] = 1**

**if a > 0:**

**if a + b <= y:**

**if get\_all\_states((0, a + b, c)):**

**ans.append(state)**

**return True**

**else:**

**if get\_all\_states((a - (y - b), y, c)):**

**ans.append(state)**

**return True**

**if a + c <= z:**

**if get\_all\_states((0, b, a + c)):**

**ans.append(state)**

**return True**

**else:**

**if get\_all\_states((a - (z - c), b, z)):**

**ans.append(state)**

**return True**

**if b > 0:**

**if a + b <= x:**

**if get\_all\_states((a + b, 0, c)):**

**ans.append(state)**

**return True**

**else:**

**if get\_all\_states((x, b - (x - a), c)):**

**ans.append(state)**

**return True**

**if b + c <= z:**

**if get\_all\_states((a, 0, b + c)):**

**ans.append(state)**

**return True**

**else:**

**if get\_all\_states((a, b - (z - c), z)):**

**ans.append(state)**

**return True**

**if c > 0:**

**if a + c <= x:**

**if get\_all\_states((a + c, b, 0)):**

**ans.append(state)**

**return True**

**else:**

**if get\_all\_states((x, b, c - (x - a))):**

**ans.append(state)**

**return True**

**if b + c <= y:**

**if get\_all\_states((a, b + c, 0)):**

**ans.append(state)**

**return True**

**else:**

**if get\_all\_states((a, y, c - (y - b))):**

**ans.append(state)**

**return True**

**return False**

**initial\_state = (12, 0, 0)**

**print("\nSOLUTION USING DFS APPROACH\n ")**

**get\_all\_states(initial\_state)**

**ans.reverse()**

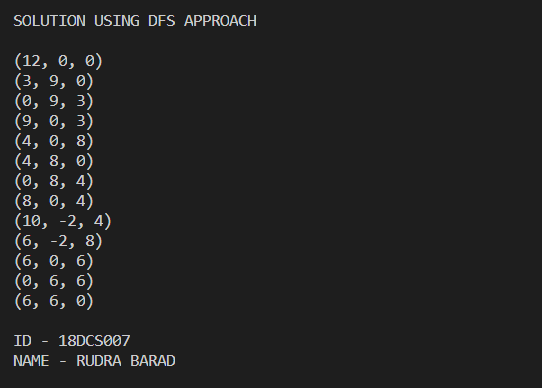
**for i in ans:**

**print(i)**

**print("\nID - 18DCS007")**

**print("NAME - RUDRA BARAD\n")**

**OUTPUT**



**CONCLUSION:**

**In this practical, we learn about Depth First Search Tree and Breadth First Search Tree and difference between them.We also solve water jug problem using both these methods.**

**3.2** Implement Travelling Salesman Problem using Hill Climbing Algorithm in Python.

**PROGRAM CODE**

**# Import libraries**

**import random**

**import copy**

**# This class represent a state**

**class State:**

**# Create a new state**

**#The \_\_init\_\_ method is similar to constructors in C++ and Java. Constructors are used to initialize the object’s state.**

**#self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.**

**def \_\_init\_\_(self, route:[], distance:int=0):**

**self.route = route**

**self.distance = distance**

**# Compare states**

**#Defines the behaviour of the equality operator ==**

**def \_\_eq\_\_(self, other):**

**for i in range(len(self.route)):**

**if(self.route[i] != other.route[i]):**

**return False**

**return True**

**# Sort states**

**#Defines the behaviour of the less-than operator <**

**def \_\_lt\_\_(self, other):**

**return self.distance < other.distance**

**# Print a state**

**#repr() compute the “official” string representation of an object (a representation that has all information about the object) and str() is used to compute the “informal” string representation of an object (a representation that is useful for printing the object).**

**def \_\_repr\_\_(self):**

**return ('({0},{1})\n'.format(self.route, self.distance))**

**# Create a shallow copy**

**def copy(self):**

**return State(self.route, self.distance)**

**# Create a deep copy**

**def deepcopy(self):**

**return State(copy.deepcopy(self.route), copy.deepcopy(self.distance))**

**# Update distance**

**def update\_distance(self, matrix, home):**

**# Reset distance**

**self.distance = 0**

**# Keep track of departing city**

**from\_index = home**

**# Loop all cities in the current route**

**for i in range(len(self.route)):**

**self.distance += matrix[from\_index][self.route[i]]**

**from\_index = self.route[i]**

**# Add the distance back to home**

**self.distance += matrix[from\_index][home]**

**# This class represent a city (used when we need to delete cities)**

**class City:**

**# Create a new city**

**def \_\_init\_\_(self, index:int, distance:int):**

**self.index = index**

**self.distance = distance**

**# Sort cities**

**def \_\_lt\_\_(self, other):**

**return self.distance < other.distance**

**# Get the best random solution from a population**

**def get\_random\_solution(matrix:[], home:int, city\_indexes:[], size:int, use\_weights=False):**

**# Create a list with city indexes**

**cities = city\_indexes.copy()**

**# Remove the home city**

**cities.pop(home)**

**# Create a population**

**population = []**

**for i in range(size):**

**if(use\_weights == True):**

**state = get\_random\_solution\_with\_weights(matrix, home)**

**else:**

**# Shuffle cities at random**

**random.shuffle(cities)**

**# Create a state**

**state = State(cities[:])**

**state.update\_distance(matrix, home)**

**# Add an individual to the population**

**population.append(state)**

**# Sort population**

**population.sort()**

**# Return the best solution**

**return population[0]**

**# Get best solution by distance**

**def get\_best\_solution\_by\_distance(matrix:[], home:int):**

**# Variables**

**route = []**

**from\_index = home**

**length = len(matrix) - 1**

**# Loop until route is complete**

**while len(route) < length:**

**# Get a matrix row**

**row = matrix[from\_index]**

**# Create a list with cities**

**cities = {}**

**for i in range(len(row)):**

**cities[i] = City(i, row[i])**

**# Remove cities that already is assigned to the route**

**del cities[home]**

**for i in route:**

**del cities[i]**

**# Sort cities**

**sorted = list(cities.values())**

**sorted.sort()**

**# Add the city with the shortest distance**

**from\_index = sorted[0].index**

**route.append(from\_index)**

**# Create a new state and update the distance**

**state = State(route)**

**state.update\_distance(matrix, home)**

**# Return a state**

**return state**

**# Get a random solution by using weights**

**def get\_random\_solution\_with\_weights(matrix:[], home:int):**

**# Variables**

**route = []**

**from\_index = home**

**length = len(matrix) - 1**

**# Loop until route is complete**

**while len(route) < length:**

**# Get a matrix row**

**row = matrix[from\_index]**

**# Create a list with cities**

**cities = {}**

**for i in range(len(row)):**

**cities[i] = City(i, row[i])**

**# Remove cities that already is assigned to the route**

**del cities[home]**

**for i in route:**

**del cities[i]**

**# Get the total weight**

**total\_weight = 0**

**for key, city in cities.items():**

**total\_weight += city.distance**

**# Add weights**

**weights = []**

**for key, city in cities.items():**

**weights.append(total\_weight / city.distance)**

**# Add a city at random**

**from\_index = random.choices(list(cities.keys()), weights=weights)[0]**

**route.append(from\_index)**

**# Create a new state and update the distance**

**state = State(route)**

**state.update\_distance(matrix, home)**

**# Return a state**

**return state**

**# Mutate a solution**

**def mutate(matrix:[], home:int, state:State, mutation\_rate:float=0.01):**

**# Create a copy of the state**

**mutated\_state = state.deepcopy()**

**# Loop all the states in a route**

**for i in range(len(mutated\_state.route)):**

**# Check if we should do a mutation**

**if(random.random() < mutation\_rate):**

**# Swap two cities**

**j = int(random.random() \* len(state.route))**

**city\_1 = mutated\_state.route[i]**

**city\_2 = mutated\_state.route[j]**

**mutated\_state.route[i] = city\_2**

**mutated\_state.route[j] = city\_1**

**# Update the distance**

**mutated\_state.update\_distance(matrix, home)**

**# Return a mutated state**

**return mutated\_state**

**# Hill climbing algorithm**

**def hill\_climbing(matrix:[], home:int, initial\_state:State, max\_iterations:int, mutation\_rate:float=0.01):**

**# Keep track of the best state**

**best\_state = initial\_state**

**# An iterator can be used to give the algorithm more time to find a solution**

**iterator = 0**

**# Create an infinite loop**

**while True:**

**# Mutate the best state**

**neighbor = mutate(matrix, home, best\_state, mutation\_rate)**

**# Check if the distance is less than in the best state**

**if(neighbor.distance >= best\_state.distance):**

**iterator += 1**

**if (iterator > max\_iterations):**

**break**

**if(neighbor.distance < best\_state.distance):**

**best\_state = neighbor**

**# Return the best state**

**return best\_state**

**# The main entry point for this module**

**def main():**

**# Cities to travel**

**cities = ['New York', 'Los Angeles', 'Chicago', 'Minneapolis', 'Denver', 'Dallas', 'Seattle', 'Boston', 'San Francisco', 'St. Louis', 'Houston', 'Phoenix', 'Salt Lake City']**

**matrix = [[0, 2451, 713, 1018, 1631, 1374, 2408, 213, 2571, 875, 1420, 2145, 1972],**

**[2451, 0, 1745, 1524, 831, 1240, 959, 2596, 403, 1589, 1374, 357, 579],**

**[713, 1745, 0, 355, 920, 803, 1737, 851, 1858, 262, 940, 1453, 1260],**

**[1018, 1524, 355, 0, 700, 862, 1395, 1123, 1584, 466, 1056, 1280, 987],**

**[1631, 831, 920, 700, 0, 663, 1021, 1769, 949, 796, 879, 586, 371],**

**[1374, 1240, 803, 862, 663, 0, 1681, 1551, 1765, 547, 225, 887, 999],**

**[2408, 959, 1737, 1395, 1021, 1681, 0, 2493, 678, 1724, 1891, 1114, 701],**

**[213, 2596, 851, 1123, 1769, 1551, 2493, 0, 2699, 1038, 1605, 2300, 2099],**

**[2571, 403, 1858, 1584, 949, 1765, 678, 2699, 0, 1744, 1645, 653, 600],**

**[875, 1589, 262, 466, 796, 547, 1724, 1038, 1744, 0, 679, 1272, 1162],**

**[1420, 1374, 940, 1056, 879, 225, 1891, 1605, 1645, 679, 0, 1017, 1200],**

**[2145, 357, 1453, 1280, 586, 887, 1114, 2300, 653, 1272, 1017, 0, 504],**

**[1972, 579, 1260, 987, 371, 999, 701, 2099, 600, 1162, 1200, 504, 0]]**

**city\_indexes = [0,1,2,3,4,5,6,7,8,9,10,11,12]**

**# Index of start location**

**home = 2 # Chicago**

**# Max iterations**

**max\_iterations = 1000**

**# Distances in miles between cities, same indexes (i, j) as in the cities array**

**# Get the best route by distance**

**state = get\_best\_solution\_by\_distance(matrix, home)**

**print('-- Best solution by distance --')**

**print(cities[home], end='')**

**for i in range(0, len(state.route)):**

**print(' -> ' + cities[state.route[i]], end='')**

**print(' -> ' + cities[home], end='')**

**print('\n\nTotal distance: {0} miles'.format(state.distance))**

**print()**

**# Get the best random route**

**state = get\_random\_solution(matrix, home, city\_indexes, 100)**

**print('-- Best random solution --')**

**print(cities[home], end='')**

**for i in range(0, len(state.route)):**

**print(' -> ' + cities[state.route[i]], end='')**

**print(' -> ' + cities[home], end='')**

**print('\n\nTotal distance: {0} miles'.format(state.distance))**

**print()**

**# Get a random solution with weights**

**state = get\_random\_solution(matrix, home, city\_indexes, 100, use\_weights=True)**

**print('-- Best random solution with weights --')**

**print(cities[home], end='')**

**for i in range(0, len(state.route)):**

**print(' -> ' + cities[state.route[i]], end='')**

**print(' -> ' + cities[home], end='')**

**print('\n\nTotal distance: {0} miles'.format(state.distance))**

**print()**

**# Run hill climbing to find a better solution**

**state = get\_best\_solution\_by\_distance(matrix, home)**

**state = hill\_climbing(matrix, home, state, 1000, 0.1)**

**print('-- Hill climbing solution --')**

**print(cities[home], end='')**

**for i in range(0, len(state.route)):**

**print(' -> ' + cities[state.route[i]], end='')**

**print(' -> ' + cities[home], end='')**

**print('\n\nTotal distance: {0} miles'.format(state.distance))**

**print("\n-----------------------")**

**print("18DCS032 - Udit Kamdar")**

**print("-----------------------")**

**# Tell python to run main method**

**if \_\_name\_\_ == "\_\_main\_\_": main()**

**OUTPUT**

-- Best solution by distance --

Chicago -> St. Louis -> Minneapolis -> Denver -> Salt Lake City -> Phoenix -> Los Angeles -> San Francisco -> Seattle -> Dallas -> Houston -> New York -> Boston -> Chicago

Total distance: 8131 miles

-- Best random solution --

Chicago -> Boston -> New York -> St. Louis -> Dallas -> Houston -> Denver -> Salt Lake City -> San Francisco -> Minneapolis -> Los Angeles -> Seattle -> Phoenix -> Chicago

Total distance: 11195 miles

-- Best random solution with weights --

Chicago -> New York -> Boston -> Minneapolis -> Houston -> Dallas -> St. Louis -> Denver -> San Francisco -> Seattle -> Salt Lake City -> Phoenix -> Los Angeles -> Chicago

Total distance: 9607 miles

-- Hill climbing solution --

Chicago -> Houston -> Dallas -> Denver -> Salt Lake City -> Phoenix -> Los Angeles -> San Francisco -> Seattle -> Minneapolis -> St. Louis -> New York -> Boston -> Chicago

Total distance: 7941 miles

**CONCLUSION**

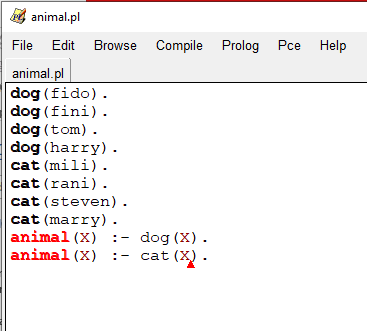
**In this practical , we learned what is travelling salesman problem and we solve this problem using hill climbing algorithm approach in phython.**

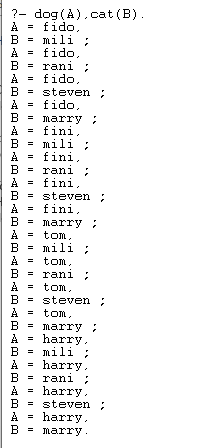
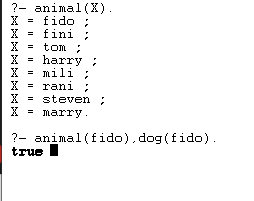
**PRACTICAL – 4**

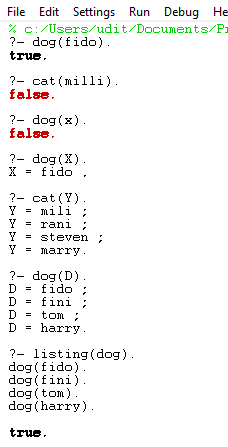
**AIM: Write Programs to demonstrate knowledge of Prolog Basics.**

**4.1 Write a program in prolog to implement simple facts and Queries.**

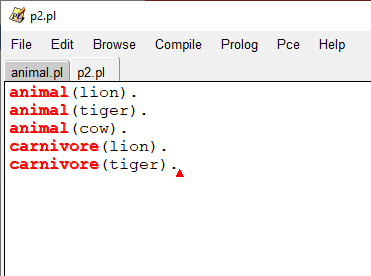
**PROGRAM CODE**



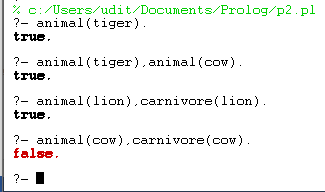
**OUTPUT**



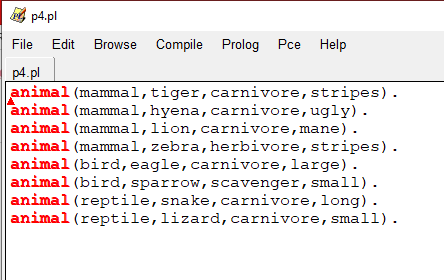
**PROGRAM CODE**



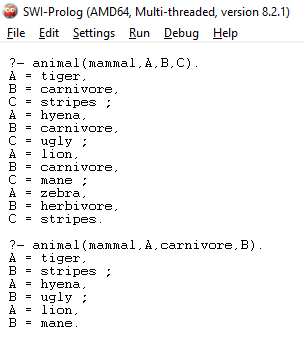
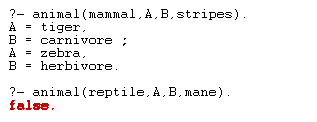
**OUTPUT**



**PROGRAM CODE**

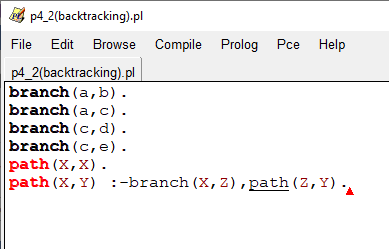


**OUTPUT**

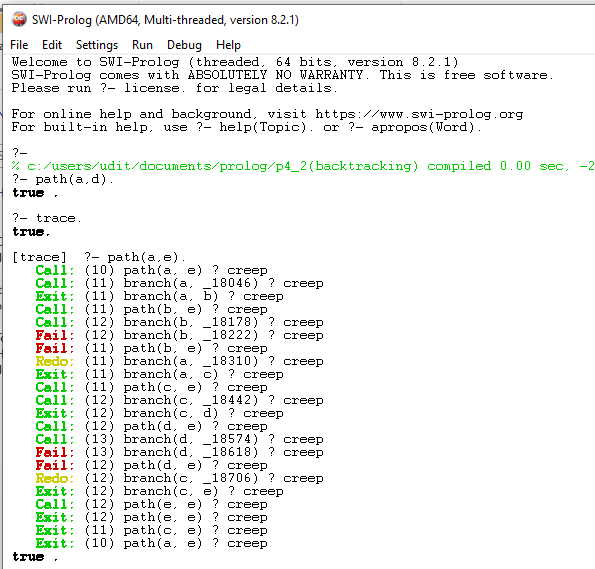
 

**4.2 Demonstrate Backtracking in Prolog.**

**PROGRAM CODE**



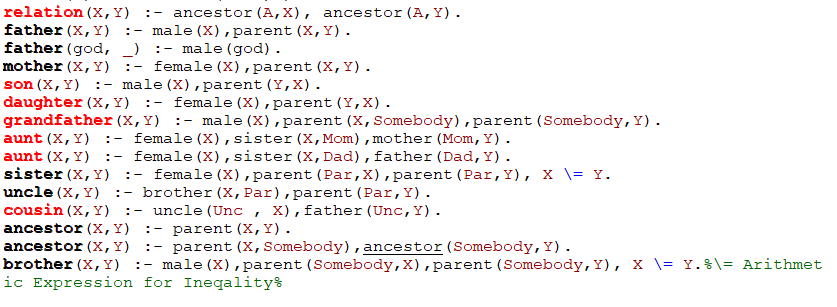
**OUTPUT**



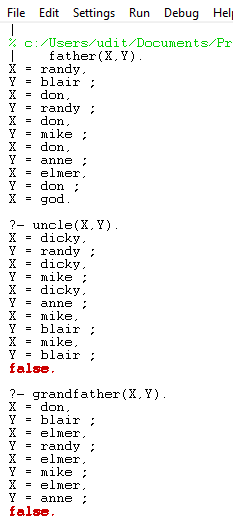
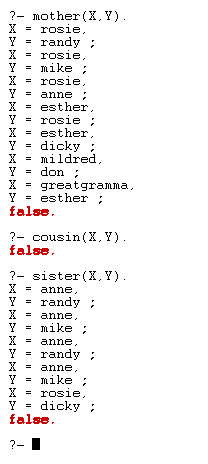
**4.3 Write a program, which contains three predicates: male, female, parent. Make rules for following family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece, cousin.**

**PROGRAM CODE**





**OUTPUT**



**CONCLUSION**

**In this practical, we learnt the basics of prolog and also the concept of backtracking. We also implemented a family relation practical which gives different relation with each other.**

**PRACTICAL – 5**

**AIM: Write a program to solve 8 puzzle problem using A\* Algorithm in python**

**PROGRAM CODE**

**class Node:**

**def \_\_init\_\_(self,data,level,fval):**

**""" Initialize the node with the data, level of the node and the calculated fvalue """**

**self.data = data**

**self.level = level**

**self.fval = fval**

**def generate\_child(self):**

**""" Generate child nodes from the given node by moving the blank space**

**either in the four directions {up,down,left,right} """**

**x,y = self.find(self.data,'\_')**

**""" val\_list contains position values for moving the blank space in either of**

**the 4 directions [up,down,left,right] respectively. """**

**val\_list = [[x,y-1],[x,y+1],[x-1,y],[x+1,y]]**

**children = []**

**for i in val\_list:**

**child = self.shuffle(self.data,x,y,i[0],i[1])**

**if child is not None:**

**child\_node = Node(child,self.level+1,0)**

**children.append(child\_node)**

**return children**

**def shuffle(self,puz,x1,y1,x2,y2):**

**""" Move the blank space in the given direction and if the position value are out**

**of limits the return None """**

**if x2 >= 0 and x2 < len(self.data) and y2 >= 0 and y2 < len(self.data):**

**temp\_puz = []**

**temp\_puz = self.copy(puz)**

**temp = temp\_puz[x2][y2]**

**temp\_puz[x2][y2] = temp\_puz[x1][y1]**

**temp\_puz[x1][y1] = temp**

**return temp\_puz**

**else:**

**return None**

**def copy(self,root):**

**""" Copy function to create a similar matrix of the given node"""**

**temp = []**

**for i in root:**

**t = []**

**for j in i:**

**t.append(j)**

**temp.append(t)**

**return temp**

**def find(self,puz,x):**

**""" Specifically used to find the position of the blank space """**

**for i in range(0,len(self.data)):**

**for j in range(0,len(self.data)):**

**if puz[i][j] == x:**

**return i,j**

**class Puzzle:**

**def \_\_init\_\_(self,size):**

**""" Initialize the puzzle size by the specified size,open and closed lists to empty """**

**self.n = size**

**self.open = []**

**self.closed = []**

**def accept(self):**

**""" Accepts the puzzle from the user """**

**puz = []**

**for i in range(0,self.n):**

**temp = input().split(" ")**

**puz.append(temp)**

**return puz**

**def f(self,start,goal):**

**""" Heuristic Function to calculate hueristic value f(x) = h(x) + g(x) """**

**return self.h(start.data,goal)+start.level**

**def h(self,start,goal):**

**""" Calculates the different between the given puzzles """**

**temp = 0**

**for i in range(0,self.n):**

**for j in range(0,self.n):**

**if start[i][j] != goal[i][j] and start[i][j] != '\_':**

**temp += 1**

**return temp**

**def process(self):**

**""" Accept Start and Goal Puzzle state"""**

**print("Enter the start state matrix \n")**

**start = self.accept()**

**print("Enter the goal state matrix \n")**

**goal = self.accept()**

**start = Node(start,0,0)**

**start.fval = self.f(start,goal)**

**""" Put the start node in the open list"""**

**self.open.append(start)**

**print("\n\n")**

**while True:**

**cur = self.open[0]**

**print("")**

**print(" | ")**

**print(" | ")**

**print(" \\\'/ \n")**

**for i in cur.data:**

**for j in i:**

**print(j,end=" ")**

**print("")**

**""" If the difference between current and goal node is 0 we have reached the goal node"""**

**if(self.h(cur.data,goal) == 0):**

**break**

**for i in cur.generate\_child():**

**i.fval = self.f(i,goal)**

**self.open.append(i)**

**self.closed.append(cur)**

**del self.open[0]**

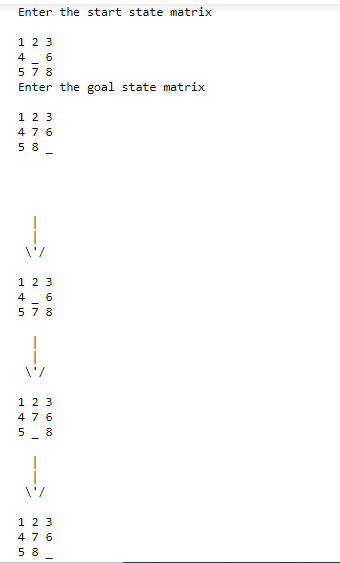
**""" sort the opne list based on f value """**

**self.open.sort(key = lambda x:x.fval,reverse=False)**

**puz = Puzzle(3)**

**puz.process()**

**OUTPUT**



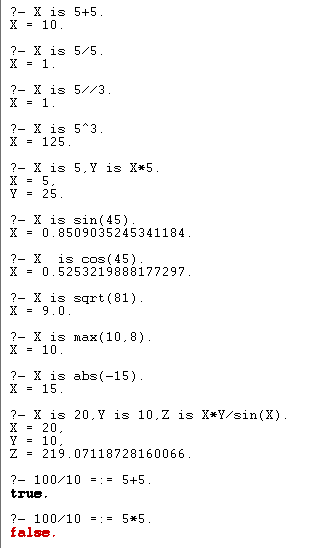
**CONLCUSION**

**In this practical, we learnt about A\* algorithm and also implemented it to solve the 8 puzzle problem.**

**PRACTICAL – 6**

**6.1: Write Programs to demonstrate knowledge of arithmetic operators in prolog**

**OUTPUT**

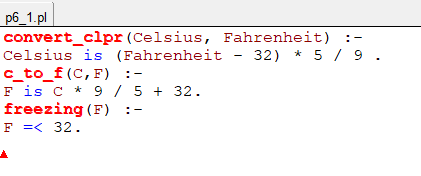


**CONCLUSION**

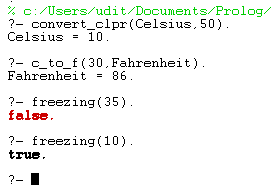
**In this practical, we perform various arithmetic operations in prolog.**

**6.2 Write a program to convert Celsius to Fahrenheit in prolog.**

**PRIMARY CODE**



**OUTPUT**

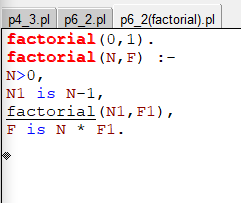


**CONCLUSION**

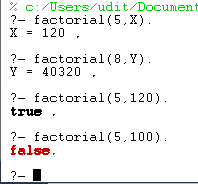
**In this practical, we implemented a conversion program from Celsius to Fahrenheit and vice-versa in prolog.**

**6.3 Write a program to find factorial of a number in prolog**

**PRIMARY CODE**



**OUTPUT**



**CONCLUSION**

**In this practical, we find a factorial of any number and also check whether the factorial of a number is true or false.**

**PRACTICAL - 7**

**AIM: Write a program to perform following operations on lists in prolog.**

**7.1 Print member of a list**

**7.2 Write list**

**7.3 Membership**

**7.4 Concatenation**

**7.5 Add an item**

**7.6 Delete an item**

**7.7 Sub list**

**7.8 Permutations**

**7.9 Append list**

**7.10 Finding nth element**

**PROGRAM CODE**

**replace([A|L],[first|L]).**

**del(X,[X|T],T).**

**del(X,[H|T],[H|T1]):- del(X,T,T1).**

**is\_permutation(Xs, Ys) :- msort(Xs, Sorted), msort(Ys, Sorted).**

**add(X, L, [X|L]).**

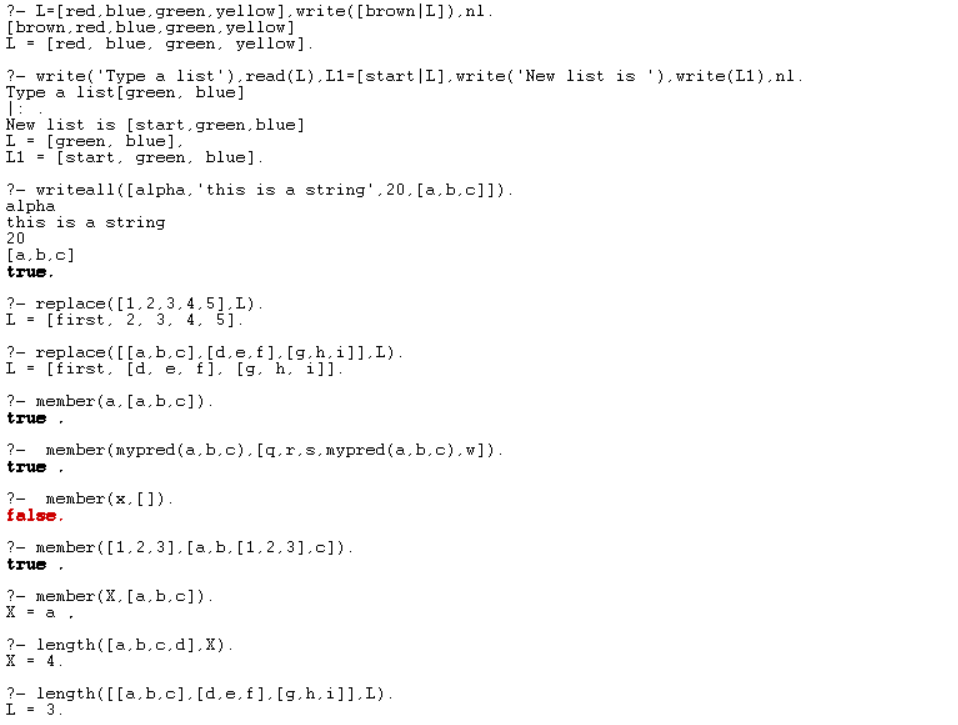
**add\_list([], L, L).**

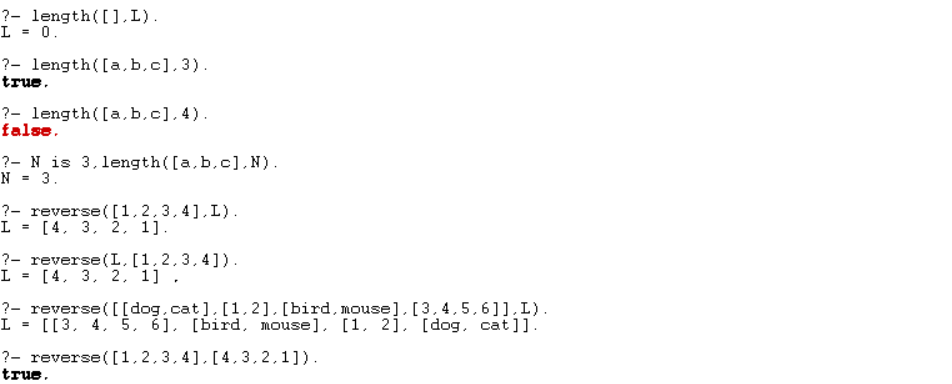
**add\_list([H|T], L, L1) :- add(H, L2, L1), add\_list(T, L, L2).**

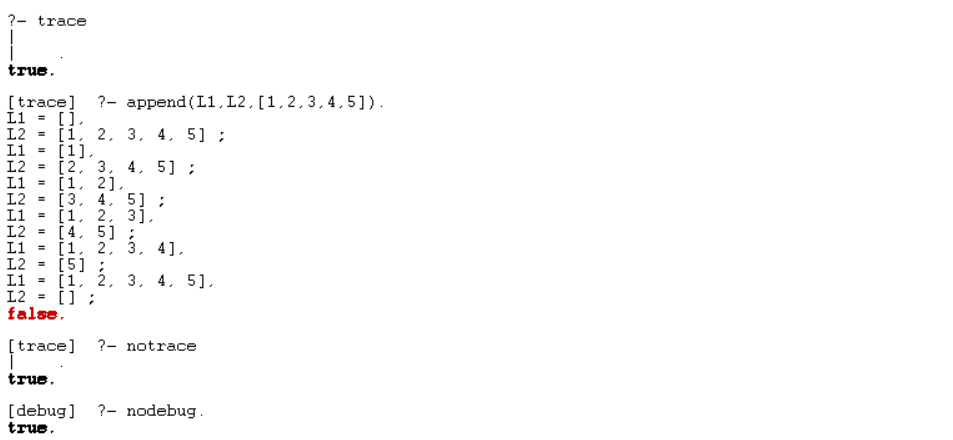
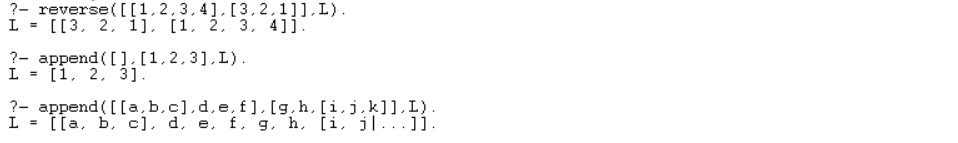
**writeall([]).**

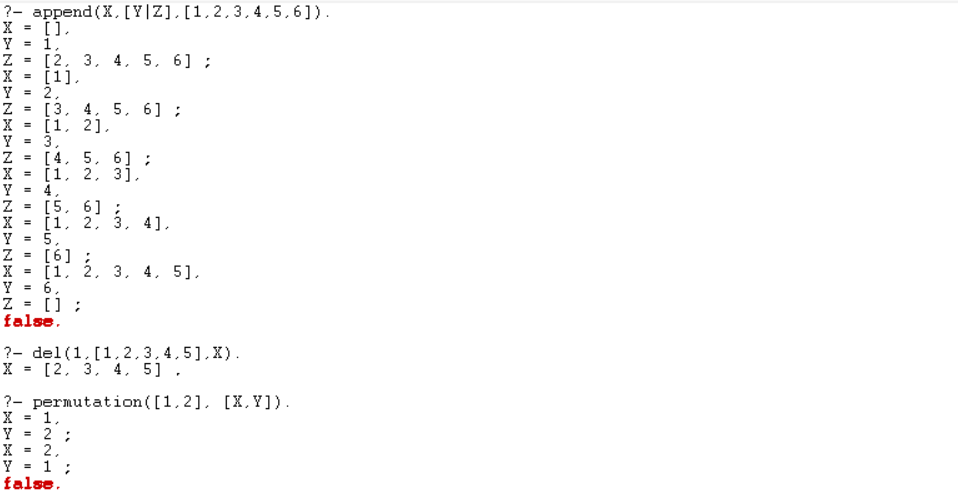
**writeall([A|L]):- write(A),nl,writeall(L).**

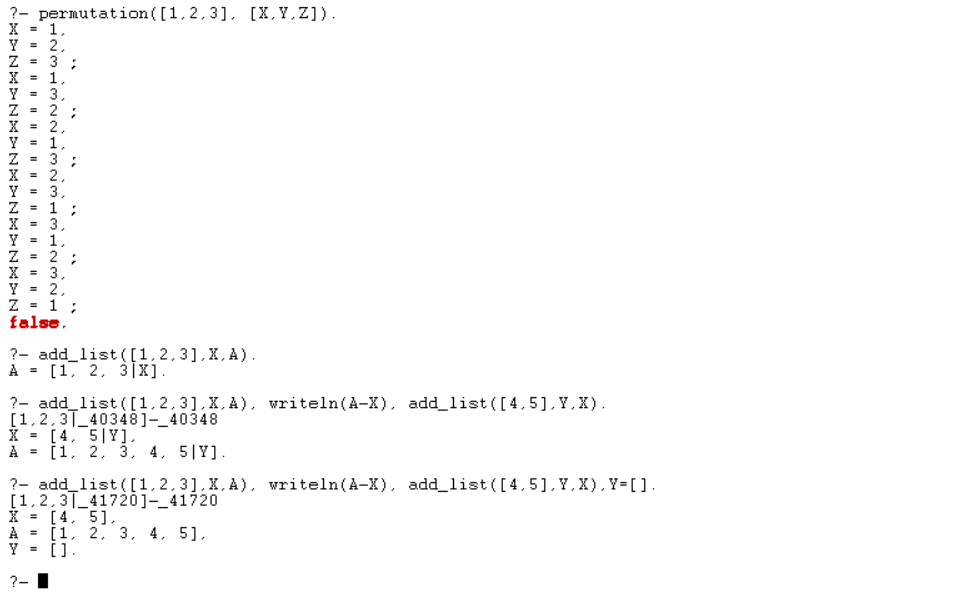
**OUTPUT:**

****

****

****

****

****

**PRACTICAL - 8**

**AIM: Perform Sentiment Analysis of movie reviews using nltk in python.**

**PROGRAM:**

**#Importing required libraries**

**import pandas as pd**

**import numpy as np**

**import re**

**import nltk**

**from nltk.corpus import stopwords**

**from numpy import array**

**from keras.preprocessing.text import one\_hot**

**from keras.preprocessing.sequence import pad\_sequences**

**from keras.models import Sequential**

**from keras.layers.core import Activation, Dropout, Dense**

**from keras.layers import Flatten**

**from keras.layers import GlobalMaxPooling1D**

**from keras.layers.embeddings import Embedding**

**from sklearn.model\_selection import train\_test\_split**

**from keras.preprocessing.text import Tokenizer**

**movie\_reviews = pd.read\_csv("E:\Datasets\IMDB Dataset.csv ")**

**movie\_reviews.isnull().values.any()**

**movie\_reviews.shape**

**movie\_reviews.head()**

**#importing and analysing dataset**

**movie\_reviews = pd.read\_csv("E:\Datasets\IMDB Dataset.csv")**

**movie\_reviews.isnull().values.any()**

**movie\_reviews.shape**

**movie\_reviews.head()**

![A screenshot of a cell phone

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDcRXhpZgAATU0AKgAAAAgABAE7AAIAAAAGAAAISodpAAQAAAABAAAIUJydAAEAAAAMAAAQyOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEhhcnNoAAAFkAMAAgAAABQAABCekAQAAgAAABQAABCykpEAAgAAAAM0MAAAkpIAAgAAAAM0MAAA6hwABwAACAwAAAiSAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAxOToxMDowNyAxMTozOToyNwAyMDE5OjEwOjA3IDExOjM5OjI3AAAASABhAHIAcwBoAAAA/+ELGGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTAtMDdUMTE6Mzk6MjcuMzk4PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkhhcnNoPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIAM0BnQMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APpGiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiub8d6/feGvDK32lR28ly97a2yrchimJZ0jJO0g9GP+B6UB0udJRXmE/wAQfEVt4suNIjtE1H+zLm2tbtbTRbs/aDLhnkWRWdIVRXU7XLFtrcjIq5p/jvVr/wARr4eeC0F/aTXx1MojgRwRY8gr8+VZxJEckn+LAHZX0uvP8B27/wBXPQ6K8W034qa6dF0uLStNe6mg0a1vri3Swvb6S7aUn92kqsxiIVG+eUvkkehrS1z4r6noXiFtMurS0xbasFvH8t18rTWSMic5YYYNMqk8j5WOPSno/wCu9v68rslXa0/rS56vRWF4L1q78ReFLXWL5Io/trSSwLEjKPILt5ROSfmKbSe2TxWtfSzQafcTWsH2ieOJmjh3bfMYDIXPbJ4pS91ajWr0J6K8fsfi9qq+F9Qm1SG2TW4ZbSEWB026gntHnbb+9tyWZ1H3gY3y4IGEJGdSPx54mlh0m1FjDBd3utvpv2q80y4tkli8h5VnSCRlcYIAKljkq2CMgh2/r7v81/VxXVr/ANf1oemUV5PY+MfEmueIPCH+mWlpFO+pxX0EVu5Sd7Zim4HzMqCOQCTtJyd2BijZfEbxKPDCPpkemp9m8Lf205vFuLhncSOpj3NLuIIX7zMSD69l/X5/5MpRb/rzS/Nns1FeV3/xO1vR4NUjv7WymuVj0+Wze2t5mWP7W7IFdAzNIU25yu0v0AUmrEfjzxNLDpNqLGGC7vdbfTftV5plxbJLF5DyrOkEjK4wQAVLHJVsEZBDs72+X5L9UTdWv/W1z0yivKtM8X+JNe8T+DVN5aWkVxLqUV/BFbMUne2fy9wJfIBHIBztJyd2Bj1Wl0uPrYKKKKACiiigAooooAKKytbTXH8j+wdQ0+zxu837bYPc7+mNu2aPbjnOc5yOmOcryPG//Qf8P/8Agin/APkygDqqK5XyPG//AEH/AA//AOCKf/5Mo8jxv/0H/D//AIIp/wD5MoA6qiuV8jxv/wBB/wAP/wDgin/+TKPI8b/9B/w//wCCKf8A+TKAOqorlfI8b/8AQf8AD/8A4Ip//kyjyPG//Qf8P/8Agin/APkygDqqK5XyPG//AEH/AA//AOCKf/5Mo8jxv/0H/D//AIIp/wD5MoA6qiuV8jxv/wBB/wAP/wDgin/+TKPI8b/9B/w//wCCKf8A+TKAOqorlfI8b/8AQf8AD/8A4Ip//kyjyPG//Qf8P/8Agin/APkygDqqK5XyPG//AEH/AA//AOCKf/5Mo8jxv/0H/D//AIIp/wD5MoA6qiuV8jxv/wBB/wAP/wDgin/+TKPI8b/9B/w//wCCKf8A+TKAOqorlfI8b/8AQf8AD/8A4Ip//kyjyPG//Qf8P/8Agin/APkygDqqK5XyPG//AEH/AA//AOCKf/5Mo8jxv/0H/D//AIIp/wD5MoA6qiuV8jxv/wBB/wAP/wDgin/+TKPI8b/9B/w//wCCKf8A+TKAOqorlfI8b/8AQf8AD/8A4Ip//kyjyPG//Qf8P/8Agin/APkygDqqK5XyPG//AEH/AA//AOCKf/5Mo8jxv/0H/D//AIIp/wD5MoA6qiuV8jxv/wBB/wAP/wDgin/+TKPI8b/9B/w//wCCKf8A+TKAOqqlq2j2OuWS2mqQefAs0c4Tey4eNw6HKkHhlBx0OOawvI8b/wDQf8P/APgin/8AkyjyPG//AEH/AA//AOCKf/5MoAv3vg7QtR1xNXvLIyXamNjieRY5GjJMbPEGCOyk8FlJHGOgqxD4c0m31y/1iGyRNQ1GJIbqcM2ZUQEKMZwMA9RgnjPQVkeR43/6D/h//wAEU/8A8mUeR43/AOg/4f8A/BFP/wDJlHSw7se3w68LNBYQf2Yyw2ECW8Ua3Uyq8SMGWOUB8TKCMgSbhyfU1a1PwV4e1i/vb3UtMSa5v7E6dcy73Uvbk52cEY57jB6c8VS8jxv/ANB/w/8A+CKf/wCTKPI8b/8AQf8AD/8A4Ip//kyjf+u+4jpLO0g0+xgs7OMRW9vGsUSAk7VUYA556CnXNvDeWsttdxJNBMhjkikXcrqRggg9QRXM+R43/wCg/wCH/wDwRT//ACZR5Hjf/oP+H/8AwRT/APyZQ9dwWmxLb/D3wzbWN3aLYSyx3kUcMrXF5NNIEjJMao7uWQKTldpG08jBqxa+DNBs47JYbJibG7a9hkkuJZJDOysrSO7MWkJViPnJ4x6DFLyPG/8A0H/D/wD4Ip//AJMo8jxv/wBB/wAP/wDgin/+TKLisrWLS+BvDyfYjFZSRNYXE1zbtFdSoyPK26TJDAsrE8qcqemMUyHwD4ags3tYtN2wvpx0tl8+Q5tslvLzuz1Y/N97nrUHkeN/+g/4f/8ABFP/APJlHkeN/wDoP+H/APwRT/8AyZQVd/19/wCepduPBfh+6W7Fxp4f7ZbQ2sx81wTHESY8Hd8rKWJDLhs4OcgUWvgzQbOOyWGyYmxu2vYZJLiWSQzsrK0juzFpCVYj5yeMegxS8jxv/wBB/wAP/wDgin/+TKPI8b/9B/w//wCCKf8A+TKd3uTZWsW4/BGgQvYvBZyQvp9zLdWzxXUqMkkrbpMkNllYnlTlT0xit+uV8jxv/wBB/wAP/wDgin/+TK0NGj8RpeOdc1PS7u38s7Us9Nkt3DZGCWaeQEYzxgdRzxgoZtUUUUAFFFFABRRRQBDP/D+NcX4r8dSeHdQntrTS/wC0BY2B1LUD9o8tooA+35BsIdzhztJUfL15rtJ/4fxrgvGPgzU9Z1K/udGmtUGraUdJu/tLspgTcSJUAVt7AOw2EqDx8wpa3Vv60dvxsUrdf61V/wALnaQzJcW8c0Lbo5FDqfUEZFR397BpunXF9eP5dvbRNLK5/hVRkn8hT7W3S0s4baLOyGNY1z6AYFQavpsOs6LeaZdFhDeQPA5U4IDKQcfnTlbWxMOnN8zGtvEmqR3Vmdd0WLTrPUJBFbSLe+bKjsCVWZNgCEgY+RpBu4z3qSx8deHdRmkjttQKiO3e5Mk9vJDG0SEB2V3UKwUkBsE7T1xUEemeItSnsYPEA02O0sJlnM1pLIz3jp9wmNkAhGcMQGk6bc45rKufh/dX2g2Gl3FxbxpFo95p87puPzzeXhlGBkDYc8g9PqDr9/5afiEVd6/13Op0bxFpuvrN/ZsspaAgSxXFtJbyLuGVOyRVbaR0bGDg4PBqnL420OO9urNbiaW4tllLLHaTMsjRjLpG4QrI690Qlhg8cGqng3w1eaNd315qUMMM1ykUQCapd37FU3HJknPAy5wqqMc/M2eKNv4Q1hdQgtZpLH+yrK/u9Qt51kczyvN5mI2QrhQpnf5gxztX5Rk4Jdbdhxt1/r+tvxLkfjhk+HMPiq+0x0Eyo62tuJpW2uwC9Ylboc/cx6Egg1Y03xzpd/qz6ZKZLe6+1vaxBoZNjsq7tpcqFVyuW8sndjnFJN4evj8NoNCha3a+gs4YQWdliZ4wv8W0kAleu0n27VHD4ZvQ0UkjWyP/AG02oyBHY/IyMuAdoy3I9OBSlpVaW3/BX6XZEb8ib3/4D/WyNPTvFOj6rqcmn2N0z3CKzANBIiyqrbWaN2ULIAcAlCQCRnrSXvivRtO1iPTLy7KXLlFOIZGjjLnCB5ApSMsRhQxBY9M1zfg/wJd+HNWtmn8t7Wwgkgt5zqd3O8oYjB8l2EUHA5ChsnGNoGKtaz4X1e81XUIrJ7L+zdWmt5rqWaR/OgMW0FUTaVbcEXBLLtJJw3SjXT+v+GKfUZbfEuwldDcWN5DE1zeW5Itp3kJgcLlYxHufIOTtB24YHoca1z428PWqWjyahviu4knjlhhklRYnOFkkZVIjQnozlRweeDihpHhjUbLXI7m6NoYILvUJoykjM7LcSB1yCoCkfMCMnsc84HNzfDDUfsNnCv2W5aTSbfTbwNql3bRp5e7LbISvnqRI3yOU6fe+Y4cdd/L8tS2o3dvP8/8AI9BsNdsNT1C8srKSWWWyfZOfs8ixq393eVClu+AScEHoRVE+NtC8m4lW4uXW3uDanZYzsZJgzKY4wEzKwKNkJuwBk8c1Z0DSZNJS/WUxn7TeyXClCT8rYxnPfj3+tczf+Bb2fQbSJGikvLLV7rUEjS+ntBKszzYXzohvRgso5AIyCOQc0L4U32X/AASVs79/8/8AgHRQ+LtDnsJr2K+Bt4LQXkj+U42xZYZwRnIKMCv3gRggHFRzeL9Ms7aSa9kb5bl7dI7WCa4lcoMn92se7gcnAIHrXOxfD67SDQ1R7SAQyyf2pEs08vmxPMJ9qySFndvMRQWYgMGkOBkKF1fwLqFzdrf2zRzXEV5dSJCup3FiGjm2dZoRvDAxj5cFTk+xCldPTz/NW/ANP69H+v6HTr4q0V7ZbiO/WSFngRZI0ZlYzY8sggcg5HzDgc5IwcR3/jHRNMTddXUp/eSRlYbWWVh5fEjFUUkIpwC+NoJGTzWCPAVxHFZQwy20USab9nuPLeU5lVZFjK7yxKjzpCSzE8L17Qat4D1KW30y4szFcX0FrNDcodVurBHeVhI8gkg+ZhvB+QjBB6rjlvfT+u39ea2F/X+f9eTOkn8Z6FBPcRNeSObaJJZGhtpZEAcKUAdVKszbhtUEs2cAGr1jrNrqelvf2PmtGm8FJoXgkVl6qyOoZT9QOCD0Nc3D4NubLRNTsraKwnE62qwRSvMilYY0XG8EvGcoSrguynDckVsaFpl/ZeHpLTUZd0zl9iG7kuhCp6J50gDyeu5hnnHQClLZ2+X9f1+QR+zf5lLR/iDoeq6OL2SdrV0soryaJ4ZOEcceWxUecN3y5TOTgdSBVhfG2htYtdLLdnbcfZjb/wBnXH2gSbd+3yNnmfd+b7vTnpWJN4N1qLQ7C20y8t7a5tNFj08tHM8W5leIsFdV3IrKjLvA3LuBAyKq2/gfWLWwvf3Fpcy3V+t2sUmt3qyQjyFjIW8wZQQVPOMMpK4UVUrXdv61/wAtSY3a1/rRfqdpPr2m22hLrM90qaeyI4nKnAViACRjI5I6jjvWfZ+OdAvr+KygubgXEsxg2S2M8XlybdwRy6ARsV5UNgsOmadf6JfX3g2HS57mO4vF8gyzS52uUkVm7c8Ke3PeqjeF7xtTu7jzYNk+txago3HIjSBIyDx97KHjpjHNT9ry/wCCv01+RX2b9f8AgFr/AITnw9m9zfsq2UMlxI728qq8cZw7xsVxKFPBKFsEj1FUtR+I2j2umvd2aXd2Yry3tZYRZTrIomcAOF8vcy4JKkDDEbQckVzdv8MdRt9IvdNRrcommXFjZ3M2qXkxkMgwpMLHy4BgDIQPnttxiug13wtqV5dX11p5tGeQ6c8EU0rIpNtOZGDEI20EEAEA/Sril7t/K/3/AOQ3a+nn+li9J430e0sUuNQuNhkmuIkS1hmuGxDIyO5VY9wVcfMxG1ScbiMEyT+N/D9vqSWMl+xld4kDx28rxKZceXulVSih9w2kkAngdDWHH4V8QaY0V3pR02e6zqEckVzNIkYjuLgzK4YISWXABXAByfmGBlbTwFc6f4fl0u2uIJAJdNMUr5Uslt5O7cAOCfKbAGRyORS0v8/8/wDgff5A/iaW2p0UHivRrnW20mG7Y3Sl1BMEixOyffRZSux2XuqsSMHI4OKeleNdP1vxGmmaUHnia1kuftTRvGrBXVRsDKN6HcSHUlTjjNYWmfD65svEgklET2Ed3dXUc51S7Z8zb+FttwhjI81gXG7IH3QWyNHwz4d1vTtS059WNh9n0zTX06E20js8w3R7ZGDKAhIj5UFsH+I54I2ur/1o/wBbEy68v9ar9COX4kWkGtXFhLYXarb6odPeXyJiW/0YzBkXy8yElSoRckgq3RlzqXHjjQLfTbW+N3NLDdxvLEtvZzTSbE4dmjRC6BSQGLAbScHB4rOHhbUj4ybUGNn9hGrjUUPmsZCDZG3KlNmAQwUg7jkE9COc7UPBWvzWFvZW93Ebcteeaianc2gjaaYukv7kAy7VYgxsVUnv3ol8KtvZfl/mW7cyS2/4P+X9XOn0fxPa61rmo6faxS7bJYnW4MTiOZZF3AqxUKfwJz1rbrmvCuhajolxKLz7K0Mlnax74pWLCSKPYw2lQNvGQ2c+wrcR70svmW9uq73DFZ2JCg/IcbBkkdR27FqT0bM43cU2WaKjt2ma2ja6jjjnKgyJG5dVbHIDEAkZ74H0FSUFBUN3dQ2VnNdXT7IYI2kkbBO1QMk4HPQVNWZ4likn8KatDBG0ksllMqIi5ZiUIAAHU1E24xbRUUnJJlKDxx4eubC6vY78i3tUSSRpLeRCyOcIyBlBkVjwpTcGPAyad/wmmiHTUvUmuXR5mtxBHYztcCRRllMATzAQBk5Xpg9CDXOnwnr+q6etzfNYWt7bQWiWUMbu0chgmWbdKdqlN5UKVAbbyct0qSfwnrc9vdXk9rp0uoX1+bqW3h1O4tRbqIliXy7qJBJnag3fIA24jjHOkla9v61/VX9CVqk/62/r1NK/8eWNneWccCNeRXj2yxvbRyyELPvKuQsZAGE4555zjAzdtvF2lSWd7PcXMcf2FHluBGsjbYxJJGGGUBYkxMMAHkYGeCcZ/CetrewXrXdrfXMQsPMeZ2i85oTL5jHCHGfMyOuSMHHWoR4Evjd6U/2m2WKO7nbUEBbM0JuTcRBTgchwoIPGGfrxTsrtef4WX63J1v8AL9X+h0i+KtHfXRpAum+1lzGuYJBG0gXcYxKV2FwvJQNuAB44NbkH3z9K8/sfAl3ZeL/tn7uWxGoS6gJZNTuywaTcdgtQRCCGc/Pk5HVcnNegQffP0qV8KvuU92lsT0UUUAFFFFABRRRQBHKhbG0ZqPyn9P1qxWbrPiLSfD4tP7ZvorU3lwltbq2S0sjEAKqjJPJGew6nAoAteU/p+tHlP6frViigCv5T+n60eU/p+tWKparq9jolkLvVLgW8BdY95Ut8zHA4AJ9yegAJOACaAJfKf0/Wjyn9P1qxRQBX8p/T9aPKf0/WrFFAFfyn9P1o8p/T9aZpup2er2ZutOm86ESyQltpX543KOMEA8MpHvjinahf2+l6dPfXrslvboXkZUZzgeiqCSfYAk0PTcBfKf0/Wjyn9P1p9rcpd2kVzEsipKgdRLE0TgH1RgGU+xAIqoNd046LFqwuP9Bl2bJdjc7mCrxjPJIHSh6bgWPKf0/Wjyn9P1qnceJNLtFna6nkg+z2811KJLeRSsUTbXfBXJGemPvDBGQc1atdRtb24uIbaXfJbFRKNpG0soYdRzwQeKAHeU/p+tHlP6frViigCv5T+n60eU/p+tWKqzanaQapbadLLtu7qOSSGPaTvWPbvOcYGN69T3oAd5T+n60eU/p+tWKpXmsWVjdrazvIbhreS5WKKF5GaOMqHICgkkF14HJzwDRvoBL5T+n60eU/p+tWAciigCv5T+n60eU/p+tWKr3F/bWl1aW9xJslvJDHAu0newUuRkdPlUnn0oAPKf0/Wjyn9P1qxUD3kEd/FZtJi4mjeREwTlVKhjnoOWXr60AJ5T+n60eU/p+tWKrPqNrHqkOnPLi7nheeOPafmRCoY5xjguvGc8/WgBfKf0/Wjyn9P1qxUC3tu2ovYrJm5jiWZ4wD8qMSFJPTkq35GgBPKf0/Wjyn9P1rP8QeKdM8MQrNrBvEhYEmW30+e4RACB8zRIwXqMbsZ7VLp/iDT9SmghtzcxzTxPNHDdWc1u5RGVWJWRVI5ZeoGc5GRQtdgem5b8p/T9aPKf0/WrFFAFfyn9P1o8p/T9asUUAV/Kf0/Wjyn9P1qxRQBX8p/T9aPKf0/WrFFAFfyn9P1o8p/T9ajudWtLO5SC5eSN5JEiQmF9rM+doDYwT8pzzxxnGRmxBOlxF5kYkC7mXEkbIcgkHhgDjjg9CORkGgCPyn9P1p8SMrEsMcVLRQAUUUUAFFFFABRRRQAVwHxR8Nx6pHomo2ukm81K21eyCzRwmSSGEThnIwDtXuTxwOeld/RR1T7NP7g6NdwooooAK47xXpmq+IvENtpth9lgs7W2kmmm1Cwe4hleUNEEULJHkhDLn5jgMOORXY0UrXGm1scla22o618NZ9Ov45YdUgiktfMlhK75oWKpMoPVWKK45PXrWDa6ZrOo3ZmisZrM36S6tmaMoUuNskUSNn7rhWhOM8eWeK9Lopy969+v8AX52fyEtEkun9flp8zxy20D7RpFzb6JoGpabayWNnDqEUlrJC092LhCzgEZkdVD7phkNkfM2OL974Zg024vrUaFJ/wi8OvxTz6bbWLyRSwmyVcpAgO9BOVYqqkZUnGQa9UooTs7/1vf8A4bsEdL/10t954Vd+HNQOjwJNplxFYY1AWltNoc9/LBM93IyMipKhhkKFCsrHA/vL36PWfByajpniqfVNFGpahJb20UEs9mJHk2wx7tn3v4xztJ5UcnFepUUrf1/Xr+Qpe82+55vDHcW2u22j2+jXyLb+ImuS6Wjrbx25gfY6yAbCMlV2g5B6gVfTT7wfCTT7M2k/2pPsu6DyjvXE6E5XGeACa7mipUEo2/rZL9Cut/63b/U8YvPCguYdV+1+HHnvLrTdait5H04uxc3bvEN+07SVbK5Izn5c5qfWNFuprPOn6a0Oii+tZJrWfRJbiJ4BZbVDWilHdVk2fKOVIBI+U49goq3ql/XVv9R395vvf8Ul+h5xJZ3Fh+z7q1vc+fldMvPKSa1NsyRneUURF3KKFICqzbgoAIBBAzE8N22p33lafoFxb6DLd2Iura4snjE86mUzSOjqN4KmNWkIKv6nFerXFvDd20lvdwxzwSqUkilUMrqeCCDwQfSpAMDA6U27u/mn9zuS9Y8vr+NjzBNHaLxuPI0W7j1pNZEq6ktsyxDTxGBsE33dm35PJznf820ferEm8Nz3FjNb2nh+9/t3+yNTt9RvGtXjW6uJAu0+Y3Eu/blWBYAcZHSvaqKhJpK3T/JL9Ne7+4adpc39dX+v3HmN+91rcms38Gk6mNOlj0xZop7GWOS5gSaQ3CCJgGb5SQUwSw4AIYZyfEugabcRi48P+Ery3ifQ9VtrIHSnDJI3llAqbS0IYGYKrBPvMAPm59korSMuWXMvP8VYUPca8jy6/wBJnfxpLIuk3ja62rWs1pqgtn2RWCrH5qefgqq4EymIkEs2dvINS+E/D0fh3UNA1NNJntpX0i7OpzR2rmSRg8LIsgUbmYDcEU5OAQo7V6ZRUrS39dLf153BJLf+v66dihq9lPqmkS21nfTWEkq/LNGOR7HowB6HaVbH3WU4I4DWPDJsItBh1rR11Ozh1OSW/lsYZ7k3SfZJFEk8beZI3zbU2l5CQF5/hHp1FG1weq1POPBXhp0160u9W0pkWzsGGnfaIc/ZY2uZjEgyPldYSi46qDitzUtJn1rV9VdrGCcRRwWlsL4MiIc+a8yEDJILR42kfNEBuUjI6uijS1vX8Xf9bB1b7/pb/I5UeE9VHhb+zP8AhJrr7RnPm/vNmMY2bvN+0bc/Nnz92f4tvyVzGueH7S11XR18Q+Gvt+nW9hfLPHp9pNdxyytJCUd0Cli7kFvm3kNlizbS9eo0UdbjWhwPgzw5dQ+Ik1DxHZGTU7bRLCBbqZQ+yQCbzQkmPvcqGIPPGavXeh3mtnVpja26zT3gitprvzEe1ijjKCVNhVy29pCNrxnbJw479hRTk+Z3ZMdEcb4j0XULf4X3um/abnV7vy0CkIzMcMvyqCWcjgn52dvVj1qPxf4at/EPieH+0dNN/Zpot4mySLfF5peEp2xv+Ule/BIrtqKlq7v/AFs1+oJWf3fg7ni8uia7ceILCfUY5zdsunPZXB0Wa5uIFVI/NUXIlRLf5xJvV+WDH7+do9fS/hkZVVLgFndButpAMocHJK8D0PRu2as0VTd7hFWSXZJfcR286XNtHPGJFSRA6iSNo2AIzyrAFT7EAipKKKQwrI8Wf8iZrX/XhP8A+i2rXpksUdxC8M8ayxSKVdHUFWB4IIPUVE480HHuVF8skzyCPSDLoV2fD2hahbacbWxXVIGtpI5L5knDXGFbBmYw7gzrnzN2AX6VLPo9u2hz+Xo9xbaE2rPLpthc+H5ryCNPJCtvsV2yIpk81l4ABOSBuGfXFVUQIihVUYAAwAKWtJO9/P8Azv8App2JWyT/AK0t/wAOeUXWhNcajp11q3hyQxW66Sxha3e88gr54YBsMW27l3HnGck96RdB1VZLe0j0+6+za9Nc2WoExHbDEl5JKGf0V4nlUN3LJ616xRT5tW/O/wCCX6E8v5frc8o07RNUj+J4ubuNhdrqk8v2mLRZd7WhVvLR73zRGYtpQeWFLBgPl43V6vRRUrSKiU9W2FFFFABRRRQAUUUUAQz/AMP41FUs/wDD+NeWfFtNOvZrHSLLS49U8WanE9vpnmMSLFCQXufSPbgEOBuJAAPBpdUhnp1FR28bxW0UcshldECtIerEDk/jUlU9yVqgorC8cf8AJPfEP/YMuf8A0U1Zvi+2ludE0FYb2ezK6nZktCsZLfOBg71b68en4ULVpeaX3scvdhzev4K519FeX6RqGraJDHd29689rc6lq0f9nGJNmUkuJVYNgPuLR4OW24PQHmnp4m1jT7MGXxAmpC40601A3BhiUW3mTojKu1QPLZWbbuyw2t8x7J6O39df8mErRbXZ2/FL9Uem0VwN3r+p3viS70e11VrKJ9aWyW6hjiZ4FFksxRd6lSzOCMsG4Jx2xhRahrmoLPqlvrxiubTSLyT7VDbxN55iupBGCpUrsIXnGGPGGHOZb5bN7f8AAuNrW39b2PW6K5Hwuk8ni7xBeTXk0gnjtG8gqgSMmLPy4UN+ZNctoum6hJqlpBba5dwSSajrJ+0+VA0kYE4GEBj28kAncp6tjHGG3a9+n+YR1jzf1s/8j1eivKrLxDreqRxTwTrHfXmlabJIbKO2juZi5uC6wmb5C3yZ2uSAocjBrro9R1Cb4dyX9lczXF/BC7BpoY45HeNjujcDKbvlKEr8pOSuBirlHlvfoDTUlHqdPRXAxeLL2a+l2X221kkkvo5fIV1js41kQjAGWBeNWznd+8wCBjGXp/iTxHINUtDfXqTtZWlxbnUxYRygyysp8kRMUBZRiNZc/PgEkZqevL1/r/g/cJa/1/XdfeepUVzmh6pLH4Vu7jUby7aaxMomk1C3iE0W0bsOID5b4BHKYBGB1zWP4H1zVr3xDeWOqz6jLH9gt7uNdRjtFcF2cZQW7HCEKMB8sMHk0dbf13/QOl/67Hd0V5/eeLNT0y+uYp7kTf2XLP8Aa4wi5dJNzWvQZBwFX3PrUN7rWvWM11JNrm2TR3sLd7RoItt+0uwOzYG4Fi7BdhUAr0YcUk72f9f1+OqB6Ox6NRXnml6xr/8AaVle3GqyXcF5qWo2f2AwRBFSEzGMqwUPu/dAcsQQemea5+417V7FbLxAupnVNQvdB85YxHCotfOuLdTsHyjYu448wnlDuYjOHre39bN/oD0bXZ2/FL9T2OivM7XWPE1xdWmkT6le2LyawbZp5xYveCH7I8u1xFviVgygg7VJXbkHkl/hXOieJZg995n9pX2otcyzQwI8phZQmWRFJ2gsec4GewAEuSjv2v8Al/mNp2X9d/8AI9JorgPh9qNxq+qtqN7L51xdeH9NlkkCgb2Zrgk4AAH4VBP4r1q3aSI3Bee2WawdAiHzLp3kEDkgcHES/LwP3vPQVpKLjLlCSs/u/FX/AK9D0aivONR8RavbeLreG31HUJbVdTg02Q/Z7RLUsyAsp3Hz2l+YtlAE6DGAxO74T1m/1e5eC7uPMfS4fst8RGq+ZdhyGOO3yoGAHBEoPpSWqv8A1tf9V95N9bf12/R/cdVRXmKS6lZeKdXs7TWrq2XVvEqWzXBigJtlFisuI9yY3NtVPmDcAcbsseh0rWtSu/h5d3hvbY39ubm3ivp9scUpikeNJW6KN20E4+XJ444oasr+Sf4J/qXJcr/rz/yOtorzg6xrYGl6Yda1KyubnV/slw+oW1obmJDbPIFzEGhJJClSBxkBgeQaVz4r8SXENlY2U19I6Jfs97ZJZrJP9nn8pXcXBWMIR8z7MHOMFRS/r8Lis/69Wv0PVKK8zOv+JLiNtWOqi1S3uNMRrCGOGWGTz/JEoL4YkfvDtKP+LDFIl1fap4t8O6hqOpsf+KgvreLTTHGqwLFDcxqQQN5JUBm3Ej5hgKOtKN/x/D/hxK7jzf1tc9NorznxKs9t441e7S7dl/s7TV+zSRRPH8144zgoTkYJHPVj6Lj0apHJWdv62T/UKKKKBBRRWd4guprLw1qd1avsmgtJZI2wDtYISDg8dRUykoxcn0HFczSNGivMp/Emt6dYmL+3lvpL21sJ/tht4lFkbicRMyAAApgllD7iCvJYcVYfVfELpeaVa6pd30lhqbW5ubQWaXtzH5KyYVZVWElGk2tgDhemc5t6Xfb9Gk/zFur/ANbXPRaK8surqfW9b0uWHVHCM2lT/aYLSKJp94nyWDqxAOMhc8ZOOtSW/iTVbOMWEM8cUmryTwab5cEa+VOt66SMAFAYiN1kIOc+Wx55otq152/BP9Sb2dvK/wCNj0+pIPvn6V5pYeJvEVz40XetzHYHVptOaKVrNbcogbBTLi4Mp2hsYKlScLjDV6XB98/SktUmU9HZk9FFFABRRRQAUUUUAQz/AMP41iap4U8Pa5dLc61oOmajOq7Flu7OOVguc4BYE45PHvXQMgb7wzTfKT0/WgCrHGkUaxxKqIgCqqjAUDoAKdVjyk9P1o8pPT9aAKc8EV1byQXMSTQyqUkjkUMrqRggg8EEdqivNOstQtPst/Z291b5VvJniV0ypBU7SMcEAj0xWj5Sen60eUnp+tAGcmnWUXl+XZ26eVI8se2JRsd87mHHBO5snqdx9ar2/h7RbS1u7a00iwggvSTdRRWyKs5IwS4Aw2R1zmtnyk9P1o8pPT9aAOW1XwVpOoaE2kWsMOl2bujyRWljbMkm0AKGSWJ0IGFx8uRtGDV3TfDum6XpqWUVskqCJoneZQzyqzFnDHHIZmJI6c9K3PKT0/Wjyk9P1oAzotOsoL1ryGzt47p41iadIlDsi/dUtjJAycDoKZBpGm2t1Jc22n2sNxI7SPLHCqszMAGYkDJJCrk98D0rU8pPT9aPKT0/WgDEn8OaHc2TWdzo2nzWzxpE0ElqjIyISUUqRjCkkgdsnFXre2gtLSO1tYI4beJBHHDGgVEUDAUAcAAcYq75Sen60eUnp+tAeZmwabY2yxrbWVvEIofIjEcSrsj4+QYHC8DjpxVS38LeH7SxlsrXQtNgtZkMckEdnGsbqeSpUDBB9K3fKT0/Wjyk9P1oAzrPT7LTrBLHT7OC1tIwVS3giVI1B6gKBgdTUGmaDpGipt0fSrLT1wRi1t0iGCcn7oHetjyk9P1o8pPT9aOtwM6TTbGZ5Xls7d2mZGkZolJcocoTxyVPI9O1Rz6Npl1qdvqV1p1pNfWwIgupIFaWIHqFcjK9T0PetXyk9P1o8pPT9aAM5NOsovL8uzt08qR5Y9sSjY753MOOCdzZPU7j61Xt/D2i2jXDWukWEDXRc3BjtkUzF8b92B827AznrgZrZ8pPT9aPKT0/WgDIstC0jTbeCDTtKsrSG3kMsMcFuiLE5BBZQBgEgkZHYmi60PSb6JYr3S7K4jSf7SqS26OFlznzACPvZJ+brzWv5Sen60eUnp+tAGda6dZWJBsrO3tyI1hHkxKnyLnavA+6NxwOgyfWg6bYmR3Nnbl5JVndvKXLSKAFc8csAoAPUYHpWj5Sen60eUnp+tAbmM3h7Rn1R9TfSLBr+TbvuzbIZW2kFcvjJwVUjnjaPSn6VpUWlQTLHLLPLcTNPNPNt3yOeMnaAOAAowBwBWt5Sen60eUnp+tAbmNdeHtFvvtn23SLC4+3bBd+bbI/2jZ9zfkfNt7ZzjtVpLG0j09bCO1hSzWLyVt1jAjCYxs29NuOMdMVf8pPT9aPKT0/WjpYLnMXHgnw9cWdjZDSrSGws53nWxht41gkZo3Qh49uCMOT9QKu3nhzRNQ0630+/wBG0+6srXHkW09qjxxYGBtUjC4HAx2ra8pPT9aPKT0/WndgZz6dZS+Z5lnbv5rpJJuiU72TG1jxyRtGD2wPSoV0PSV1V9TXS7Iag5DNdi3TzWIUqCXxk4UkdehIrX8pPT9aPKT0/WkHkZdxpOm3d2t1d6faz3Cp5YmlhVnC7g23cRnG4A49QDT006yjZWjs7dGV3dSsSghnOXPTqxOSe/etHyk9P1o8pPT9aAKdvbw2ltHb2sUcMEShI4o1CqigYAAHAAHapKseUnp+tHlJ6frQBXqvqFlHqWm3NjOWWK5haFyhwwDAg4z35rQ8pPT9aPKT0/Wk0mrMabTujEsvDmjafp09jaaXZx211n7TGtugW4JGGMgAwxI6k9aZJ4W8PzaPFpMuhaZJpsL74rJrOMwo3PITG0H5jzjufWt7yk9P1o8pPT9ab13EtNjKuNG0u7Ui6020nDeXkSQK2fLOU6j+EkkemeKeNMsFaBlsbYG3keWEiFf3Ttncy8cE7myR13H1rS8pPT9aPKT0/WgDJGi6WNYOrDTbMakU8s3ogXziv93fjdj2zWlB98/SpPKT0/WlVFU5UYoAdRRRQAUUUUAFFFFABXGfEWfxBpmjyazoesPZQ2MReSFbWOSMkEEyTMwZ/KVQcrEu/nIJ7dnXO+JfB0Hidyt3quqW1pLAbe6sradRDdRFslXVlbaTyNyFWIJGemF1GrdTfhlWe3jljZXSRQysvQgjORT6REWONURQqqAFA6AUtU7X0JV7ahRRRSGcr8QdVu9J0Oyksr27sTPqMEEs1lbC4mEbNhtkZR9x+ik1lWHi+703w/qt1c3q30drexxQXGvSR6W6RuqndcZjUxjcWC/ugzfLhSDuPVeIdBTxBYw273t1Yvb3EdzFcWvl70dDkcSKykexBrKl8CQzw+ZPrWqSamLpLpdUbyPPRkRkUBRF5W0K7jGz+InrzUxur+v4af5MOvy/HX/gGSnxPM+j6beW1npe68lni3z6wsVtI8Unl7IJ/LIldzygITIBORitF/Fl1HqV/YWdp9qv21QWVrBdXKxRjFqk7EusZKqAW7SEseuD8sx8CwLp5srbWtXt4JJJ2uVWWNxciZy8iuHRhySeV2sASA1TXHgqykaaW0vb2wuXvFvYrm3aMtbuIVgwodGUqY1wQwbqTwcYtW1v/Wq/QO5k+IviJN4be0XUtP061eSFJJoLvWY452LPtZbeMKxmIxnnYDlQOcha3/CYaraazMZSbu2t49Tle3AVCVguIVUghc5VHfjv39RqXnw7tLzzg+taysdzbJb3aCdGN1sLFXd2QuCC5OFKqcAFSvFXD4UgsrmbUdPDXN75V2I4LqYJC5uHR2DFUJAygAIBwCeDSWkrvtL/AIA12fl+tzQs9VfUP7S+xwKwtJTDDI8mFnYIrHkA4AZtueeVPFefaV488Tw6Fa6lf2MGoCPQI9TvEW5EY2hn3OpEXMjKARHhV4I3Dqe98L6EnhvwvY6Sr+Y1vHiSTn95ISWduSTyxJ5JPNZtr4B0+10W40z7bfTQzaadL3SNHuSDL4AwgGQHIBIPAGcnJJa0v67P9bFRcWrS8vuv/kZV58VNOtfFw0f/AEERpdwWcol1FY7vzJlUq0duVy8Y8xAW3DHzYB28tuviXJZaWt7e2GnWCXWozWNk9/qvkRP5TSK7yuYsRg+WdoG8sWA461v/APCIwrrH2221XU7WF5Y5p7K3mVIbiRFChmO3eOFUFVZVbbyDk5JfB9qdMt7W0vr2ymtbqW6t7yBozLE8jOzgBkZCpEjLhlPGO4Bp6cvn/wAN/wAH5GetzGg8bTzeGX8WW7Wsum3EcUMVvJcqsNvMs0iTSPcKrDyvu/PgjC5wM10Vhrcl34TGsSW0LN5LyiO0vI5o5QucGOY7VKsACGbbweQOQBtBb/hH4dMh1jVIZYSrLqAnV7gsDksxdSjZ5ypXbg4AGBiP/hFLB/CN34euJJ57a9SZbmZ2USytKWMjkqAoYlieAAOwA4pPr+BS6N/1/X9edDwZ40XxXNfwY0xpLIRMZNK1MX0DBw2B5gRMMCpyuPQ5OawYvGetaRZ39/rkmkqJdVmtbddR1mK1tYEj3DaJPIDlmx0IfOCcr0rr9E8Nro99dXsmp32o3V1FHFLLeGPkRl9uBGiqOHI4AHAOM5Jpz+CbSS7gurXUdQsrmG4uJhLA0eWWdg0kZ3IwCkqvIAYY4Yc0pX5vd7fjp/wRR297v+j/AOAYj/E26l05b/TtAFzax6Xa6ncFr0IyxzFwVQbCHcbMjJUHnlcc2v8AhMdQkvobG9sf7PvY9SezmS2ulmiP+htcKdzRAkY28AKQR1IyDet/AGlWulT6fFPeeTNp8GnEtIpYRQlyhB2/e/eHJOeg/G1N4QsJ9XfUXmuRM959sKhl27/sxt8fdzjYc+ue+OKuXK7287FRspa7W/G3+Zl+G/GNze6Pp0V9beZqM4tEB8wDz1lgErTYC/KBiXjGMx4zzxpa94ofRLmeAWH2iQWiz2yibaZ38wRmP7uFwWjOec7unHMGl+EU0/xJaXoC/Z9N0xNOtG80tJKB/HINoUFQCBjP336ZxWrqWhWuq6hp95ctKHsJGkRUYBZMjG1xjkZCtjjlV+hJNN3Xd/1+v4GcLqNn2X9fp+JzN54w1a58SW1t4dsYrqFjOirNceVHcFYoZN5fYzIB5jKMA5PXA5GdqXxl02wt7Ocw2cKyadDqVxFfaklvMIpCcLAhU+dIAjErlR90Zy3HXab4T0/Sjp/2V5/9AWZY9zA7vNILFuOegxjFZ6eALS3trSDTtX1bT0gtIrKY2s6I11DGTtDtsJUjc3zRlG+Y88DExtfXy/4JatbXcp6l8Q30xNaubjT7WKw027isYrm4v/KE00ixspbKbY4gJRl9xI2nCmtfwh4sg8WaRcXdv9leS1uGtpvsN2tzCzgBsxygAMCrLyQpzkEDFSXPhSyubXUIfPuonvbtb3zo3AeCZVQKyEgjjy1OGDA8g5BxVvTdHFjpctndX97qZnZ2mnvZAXfd2wgVVAHACqo9skkn2fO346frf8Cdb69/8/8AgfiYHhTx6viTXrjTGi01ZI7f7QBYaql40Q3bSkwVQI5BkcBnHXDHHOP4g8X6zYr4kjspt09ot61puKKkQitoHGf3bFiDIxHPJ4ORjHS+H/BkHh+7guF1XUb97ez+xQi7aPbHCCpVQqIo42/e+8c8k4XBeeB9Mvri/lnlus363CyqrqABNFHE+Pl4+WJce5PXsS207fr/AJDjpNOW1191tfxOf/4SbxOt9dJfLaW0cd5pcS+RP5xAmZBKvzRLwcnnr6AVa8K/E+w8U+II9PtvsBS5hkmtjb6ik0yqjAETwgAwsQwIGW6HJBGK1h4LtjMZJ9Sv5yzWruH8oB5LdgVc7YxycAMBgY6AHmptG8KxaJdh7fVNSltIkaO1sJZV8i2UnOFCqC2MADeW2jgYoX6v/gEpP8F+tzHufFmrad4l1lH09brSLG4tkknM4jaBZUQfIgU+YQxLNuZcAjbuPAfF45vJ9QAi0ZG0+5urmxsbkXnzzXEIfKvHswiMYpAGDMeBlRni/e+CrO/1i4vpb6+WO7lilurNXTybgxAeWGyu4AFQflYZ6NkcUlt4Ks7PXBqMd5eyxxTzXVtp8sifZ4J5c75Fwm/J3PwWKje2AOMJba/1/XT8fLTT+vn/AMAt6J4hTXJsW1uywizguGkLfdaUMfLx6qACf94Vs1heEtBfQNKnW4WFbu9upbu4WB2dEZzwiswBIVQqg4H3eg6DUS1mRlLX9w4DuxVljwQx4U4XovQd/UtVMks0VHbxPDbRxSTyXDogVppAoaQgfeO0AZPsAPapKQBXIa/40vtH1i/trTREu7XTbe3ubu4e88phHK7qdi7CGZQhOCyg56119Y2o+F7LU31Rp5Z1Op28VvNsZRtWMsQVyOD85znPagUr20Ocj+Kemv4wOjlrAQi8lsSRqKfa0ljVizNbYyI8owDbic4+XBzVjT/HV/cqJbvQUghuNMl1S0KagjNJEpTar71RY3IcE/MVX+8eca0XhSK31h7y31LUEtmme4OmLKq2zSuDuY4XeQSSSpYpk525rmNH+HmoTLdW/iO4MdidPk060t4L/wC0tBE7KQFcwRY2bFA3K7H+JiABSjfr/Wj/AFt8txy393v+v+W/nsRw/ETU9aa3i0aPRZZItUtre5ksdX+1QPFIrHAkEP3sqcjaMcc81o23j5orO9E1nJLPGHa0Es6g3T/a5LcRZVAFwwjGcE4cZyQSdBPAsIke4m1rVbi9kmt52u5Wh3loS23CiMIAVYqQFAxzwxLGQeBtK+1aVcGS5L6XdT3MX7wASGVy7K4A+ZQ5VgOxRTziqVk9e/4WX63J978P1f6aGTbfE+wufGi6En2Aq17JYALqKG7WZASS1sVBEeVZQ24knB24Oa7qsK38KxWutm+ttU1KK2M73J01JVW3Mrg7nOF3nJJO0uU3HO3PNbtJfCr7lPd22CiiigAooooAKKKKAGSSbMcZzTPP/wBn9aJ/4fxrkfHfji18D6Vb3Mtu15c3M6RQ2yNtLAsAzk4OFXcMnGMlR3FHWwdLnXef/s/rR5/+z+tRUUAS+f8A7P60ef8A7P61CTgE1xmieNdV1C10XUNS0axs9P1l1S3eHUmmmVmRmXdGYUHRTnDHHXkUX1sOztc7nz/9n9aPP/2f1rjdR+I+iWuj6he2LT3r2UQmES2sy+ehbbvjbYfMTPV03KMjJ5FS23jzTZbjWEuobu2j024igDPZz7pzJGrKETy9xbLEbF3NgA9CKBdWux1vn/7P60ef/s/rXOxeM9DnfT0t7qWaTUd/2eOK0lZvkcI+9QuY9rMA2/bt5zjBw278Tiz8b2egS2o8u6t/MW6MuMSfOVj2Y5ysbnOe2Mc0dUg6XOk8/wD2f1o8/wD2f1rkLLx3aSvqz31u9tb2N0ttA8QeeS73ZAKxIpbJIbAG7gZ6dIL/AOJGmW0tt9jtry8huLO7uQ6Wk4KNAQDGy+WSpyWB3YKkDI+YUuZWv/W1xpNuyO28/wD2f1o8/wD2f1rl4PHOivHp/wBpmmtpb2KKQI9rNth8zhBI+zEZY8Lv27u1PPiuKGS5lu4dlhbeasl3EXkxIkojEQTYGZ2J4C5yeBuzT68vr+G5N7q/p+J0vn/7P60ef/s/rWKuux3vh261PRree8eGOQpayRPbyvIgP7sq6hkJIA5XuD0qtB4jnfxJaaTc6abdrmO6lV2lzhYZEQHG3+IPnrxjvmnZ3sV0udH5/wDs/rR5/wDs/rWFf+K9G0zVo9Nvbsx3L7MgQuyRb2Kp5jhSse4ghd5GT0zUq+I9KZYW+2KFmnmt0LIww8W7zAcj5QvltknA468jK6XEbHn/AOz+tHn/AOz+tYujeJdK195k0yeR3hCs8c1vJC2xs7XCyKpZDg4cZU4ODxWfdeO9F07zF1K4ZZFefEdrbT3DbIWCuxCR5GMjPBAz1I5o62Dpc6rz/wDZ/Wjz/wDZ/Wud8ReKrbQNCtdVEMt7Bc3FvCn2aN5MrK6rvARWJwDkccnCjkiodO8baXe6rJps7NbXYupraNWjcxyNGTx5m0IHKqW2Z3Y5xjmjy/rS3+aDpc6jz/8AZ/Wjz/8AZ/WsPSPFGka7cywaZcvJJGgkAeCSISITgSRl1AkTI++mV6c8isu/8ReI7Xxdb6LbaHpcyXUUs8M8mrSIfLjZA25RbnDfvBgAkcHmjrYDsPP/ANn9aPP/ANn9a5SXx9odlBu1K78t/wB65W3gmnCxxytG0jYjyqgr8zEBVJ6kYJsx+NNClsbq7ju5DHayJG6m1lDsz/c2IV3SBs/KUBDds0dLgdF5/wDs/rR5/wDs/rWbpusWOraaL+xnD22WDM6mMoVJDKysAVIIIIYAjHNc5qHxJ0eLQLrUNJ86/kh8rZAbW4TzBI21JBiJmMZ5xIqspxgZOBSbtuG52vn/AOz+tHn/AOz+tct4h8VyeHbfRpbmwWT+0LlIbjbOVFspQszjcoLBcHghTjPQ8U2bxnBb+NLjRbiGOKztbB7ufUJJwFjZChZCuOMJKjbs9+lVbW39aagve2Or8/8A2f1o8/8A2f1rmF8d+Hm05r03c0caTJAY5bKdJt7jKL5TIH+b+H5fm6DNEnjvw/HpcOoG7neGZpFVI7Kd5gYziTdEqGRQp4YsoAyM4yKQLXY6fz/9n9aPP/2f1rHHiLSWjlkW9R0iMIZlBIPnYERXA+YMSACMjP0pdK1G9vjJ9t017EKiMu9924nOR0HTA/OjYN1c1/P/ANn9aPP/ANn9a5m38aaS4skmmbzrtEcfZ7eaWKMOxVN8nlgIGIIG/bnFdDRurh5Evn/7P60ef/s/rUVFAEvn/wCz+tHn/wCz+tRVDd3UNlZzXV0+yGCNpJGwTtUDJOBz0FJtJXYblvz/APZ/Wjz/APZ/Wuag8ceHrmwur2O/It7VEkkaS3kQsjnCMgZQZFY8KU3BjwMmnf8ACaaIdNS9Sa5dHma3EEdjO1wJFGWUwBPMBAGTlemD0INN6bgdH5/+z+tHn/7P61yF/wCPLGzvLOOBGvIrx7ZY3to5ZCFn3lXIWMgDCcc885xgZu23i7SpLO9nuLmOP7Cjy3AjWRtsYkkjDDKAsSYmGADyMDPBLs9RXV7HRef/ALP609Jd7Yxjj1rn18VaO+ujSBdN9rLmNcwSCNpAu4xiUrsLheSgbcADxwa3IPvn6Uulxk9FFFABRRRQAUUUUAQz/wAP41578QPh1ceLFuLvTdZubW+khht1hkMRgEazLI3WJnUnaD8pGSqZ4FeiSoWxtGaj8p/T9aOqYEMSNHCiPI0rKoBkcDc59TgAZPsAKfT/ACn9P1o8p/T9aAIzyDXJ+DvAOleG9Aso30rS49XjtfJuL+1tkEkhIwx8zaGOffrXYeU/p+tHlP6frSsh3aR5XYfDTU7PR7myAtN8GltYWc0mq3k/m5KkEpISlup8tcqiv2wQFwdPWPBWq6rdX1w4tlZ9Qt9SgSLUJ4CzrbiF4jLGgdBjJDrknPKgZB9B8p/T9aPKf0/Wqbb3/rW/5iWjduuhxfhzwhd6TqmnXtwLZPJt7xZo0uZ7hhJNLE4xJMSz4EZyx25J4UZwH+KfDGqard3d9o9xawXgtYBZPPuISeOVnywA+6Qdpxzgmux8p/T9aPKf0/Wk0nby2/H/ADHc8/ufBOrWlg9tot2pi3WimM301q9xFFEUZWnjUvGS21srnO3B4JqPT/AurWFvaxCa1lCpqcUrPcyuyrdOHRgzhmcgqAdxBOc57V6J5T+n60eU/p+tJq6afW/4hGTja3T/AIP+Z5nL8P8AVbi7hlmjtWFzFareAaveRxwmIBTtii2LOCFBG/YQeu4cV0Uvh28/4RrU7FUs7ma6vpbmOOaSREKtLvA3p80b46OoJVsMAcV1XlP6frR5T+n60+t/X8Wn+gl7trdLfgYvhqwvtN0cQanKWl8xmWM3Ul15KnonnSAPJ67mGecdAKoa9pmtt4q0zWdCtrC6+y2txbyw3l29vnzGiIIZYpM48s8YHWup8p/T9aPKf0/WnfW/9dgWhwuq+Gdf1O4vYwNNhtdaS3+3kzyO9q0fDiL5AJAwAAJ8vaecHpTp/Akt5rWvtc3Ua6bqNpNFapGDvgkuFVZ2I6Efu0K98s+e1dx5T+n60eU/p+tTZfn+I02ndeX4HG+DPC17ol7cXOpRwo7W8duhTVLu9ZwpJLEznCAk8IoOOfmOeEtfCN7DrVxePJbFZIb+NMMdw+0So6/w9MJz74612flP6frR5T+n60NXd/Jr79xbK3o/u2OV1Hw/qEvgXTtLszbPfWJsnAllZIpGgkjcjcFJAOwgHaeo4qvJ4Qu5o7dHlgVV1m4v5SrHJjkjmQAccsPMXrxwea7Lyn9P1o8p/T9aqT5m2+t/xt/kgj7kVFdP8rfqcJ4J8F3vh7UElv1hK2lkLOCVdUu7ppVyMt5cp2QAhF+RQ3P8WBzv3ekXE/jTTdXR4xb2lncQOpJ3lpGiKkDGMfuznn0rc8p/T9aPKf0/WnzNu/8AWv8Aw4kktjgf+EH1D+zdVg8208y8069tIm3N8rTTySLn5eBh1zjuOhqTXPBN5qc1zPHLGX3WcsEYupbcu0IcMpljG+PIfhlyR3BHB7ryn9P1o8p/T9ai2lv63v8AmP8Ar70l+SOY0Lw1LZeErvTL3ZBLfNM0vk3U9zs8zI/1szF3IGMt8uT0ArFTwhr99CG1Z9NintrS3soBbSOyyrHMkjyMWQFSdgwg3Ac/Mc8eg+U/p+tHlP6frRZXv/XVfk2Lpb+t7/mkc94l8PHxDJp8cnlG2heb7Qj5yyPBJFgDHJy4644zXMxfD7WLnTQmrajavf3OnXsF5cxBiDPO8RQqvGUVYgvJBwBXo/lP6frR5T+n61SbTuhnDReFtZv9aGs6yNPgumuLPdb20ryxrFB5hzvZFJYtKeNoAAHJ61S1bwFqU94l/aGGe4Sa9xCNUubEGOeRXBMsA3EjYAUIIOeowM+jeU/p+tHlP6frU2V7/wBa2X6B/X3Xt+Zwdt4cktvEug2FujJZ6bYR/bSkEggkaLi3VXdmJwzSNjcxG1dxGRu7g1J5T+n60eU/p+tU23uBwFr4P1qzk0hYDaRNawxRzXsF9PE4CsSytCF8u4UgkDft27mIzXZo96WXzLe3Vd7his7EhQfkONgySOo7di1XPKf0/Wjyn9P1qUrKwEFu0zW0bXUccc5UGRI3LqrY5AYgEjPfA+gqSn+U/p+tHlP6frTAZWZ4likn8KatDBG0ksllMqIi5ZiUIAAHU1reU/p+tHlP6frUyjzRce5UXytM88PhPX9V09bm+awtb22gtEsoY3do5DBMs26U7VKbyoUqA23k5bpUk/hPW57e6vJ7XTpdQvr83UtvDqdxai3URLEvl3USCTO1Bu+QBtxHGOe/8p/T9aPKf0/Wrbvdf1q7v8SVorf1tY4Z/CetrewXrXdrfXMQsPMeZ2i85oTL5jHCHGfMyOuSMHHWoR4Evjd6U/2m2WKO7nbUEBbM0JuTcRBTgchwoIPGGfrxXf8AlP6frR5T+n60X1v53/CwrI4Cx8CXdl4v+2fu5bEahLqAlk1O7LBpNx2C1BEIIZz8+TkdVyc16BB98/Sk8p/T9afEjKxLDHFJaKw3q7slooooAKKKKACiiigAoqh/Y9t/z1vf/A6b/wCLo/se2/563v8A4HTf/F1PvEXl2/r7i/RVD+x7b/nre/8AgdN/8XR/Y9t/z1vf/A6b/wCLo94Ly7f19xfoqh/Y9t/z1vf/AAOm/wDi6P7Htv8Anre/+B03/wAXR7wXl2/r7i/RVD+x7b/nre/+B03/AMXR/Y9t/wA9b3/wOm/+Lo94Ly7f19xfoqh/Y9t/z1vf/A6b/wCLo/se2/563v8A4HTf/F0e8F5dv6+4v0VQ/se2/wCet7/4HTf/ABdH9j23/PW9/wDA6b/4uj3gvLt/X3F+iqH9j23/AD1vf/A6b/4uj+x7b/nre/8AgdN/8XR7wXl2/r7i/RVD+x7b/nre/wDgdN/8XR/Y9t/z1vf/AAOm/wDi6PeC8u39fcX6Kof2Pbf89b3/AMDpv/i6P7Htv+et7/4HTf8AxdHvBeXb+vuL9FUP7Htv+et7/wCB03/xdH9j23/PW9/8Dpv/AIuj3gvLt/X3F+iqH9j23/PW9/8AA6b/AOLo/se2/wCet7/4HTf/ABdHvBeXb+vuL9FUP7Htv+et7/4HTf8AxdH9j23/AD1vf/A6b/4uj3gvLt/X3F+iqH9j23/PW9/8Dpv/AIuj+x7b/nre/wDgdN/8XR7wXl2/r7i/RVD+x7b/AJ63v/gdN/8AF0f2Pbf89b3/AMDpv/i6PeC8u39fcX6Kof2Pbf8APW9/8Dpv/i6P7Htv+et7/wCB03/xdHvBeXb+vuL9FUP7Htv+et7/AOB03/xdH9j23/PW9/8AA6b/AOLo94Ly7f19xfoqh/Y9t/z1vf8AwOm/+Lo/se2/563v/gdN/wDF0e8F5dv6+4v0VQ/se2/563v/AIHTf/F0f2Pbf89b3/wOm/8Ai6PeC8u39fcX6Kof2Pbf89b3/wADpv8A4uj+x7b/AJ63v/gdN/8AF0e8F5dv6+4v0VQ/se2/563v/gdN/wDF0f2Pbf8APW9/8Dpv/i6PeC8u39fcX6Kof2Pbf89b3/wOm/8Ai6P7Htv+et7/AOB03/xdHvBeXb+vuL9FUP7Htv8Anre/+B03/wAXR/Y9t/z1vf8AwOm/+Lo94Ly7f19xfoqh/Y9t/wA9b3/wOm/+Lo/se2/563v/AIHTf/F0e8F5dv6+4v0VWtrCG1kLxPcMSMYluZJB+TMR+NWaav1KV+oUUUUxhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAf/9k=)

**#distribution of positive and negative sentiment in our dataset**

**import seaborn as sns**

**sns.countplot(x='sentiment', data=movie\_reviews)**

![A screenshot of a cell phone

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDcRXhpZgAATU0AKgAAAAgABAE7AAIAAAAGAAAISodpAAQAAAABAAAIUJydAAEAAAAMAAAQyOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEhhcnNoAAAFkAMAAgAAABQAABCekAQAAgAAABQAABCykpEAAgAAAAMyMgAAkpIAAgAAAAMyMgAA6hwABwAACAwAAAiSAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAxOToxMDowNyAxMTo0MjowNgAyMDE5OjEwOjA3IDExOjQyOjA2AAAASABhAHIAcwBoAAAA/+ELGGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTAtMDdUMTE6NDI6MDYuMjE2PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkhhcnNoPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIASQByQMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APpGiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACisEeOvCTAEeKdFIPII1CLn/x6l/4Tnwn/wBDRov/AIMIv/iqAN2isL/hOfCf/Q0aL/4MIv8A4qj/AITnwn/0NGi/+DCL/wCKoA3aKwv+E58J/wDQ0aL/AODCL/4qj/hOfCf/AENGi/8Agwi/+KoA3aKwv+E58J/9DRov/gwi/wDiqP8AhOfCf/Q0aL/4MIv/AIqgDdorC/4Tnwn/ANDRov8A4MIv/iqP+E58J/8AQ0aL/wCDCL/4qgDdorC/4Tnwn/0NGi/+DCL/AOKo/wCE58J/9DRov/gwi/8AiqAN2isL/hOfCf8A0NGi/wDgwi/+Ko/4Tnwn/wBDRov/AIMIv/iqAN2isL/hOfCf/Q0aL/4MIv8A4qj/AITnwn/0NGi/+DCL/wCKoA3aKwv+E58J/wDQ0aL/AODCL/4qj/hOfCf/AENGi/8Agwi/+KoA3aKwv+E58J/9DRov/gwi/wDiqP8AhOfCf/Q0aL/4MIv/AIqgDdorC/4Tnwn/ANDRov8A4MIv/iqP+E58J/8AQ0aL/wCDCL/4qgDdorC/4Tnwn/0NGi/+DCL/AOKo/wCE58J/9DRov/gwi/8AiqAN2isL/hOfCf8A0NGi/wDgwi/+Ko/4Tnwn/wBDRov/AIMIv/iqAN2isL/hOfCf/Q0aL/4MIv8A4qj/AITnwn/0NGi/+DCL/wCKoA3aKwv+E58J/wDQ0aL/AODCL/4qj/hOfCf/AENGi/8Agwi/+KoA3aKwv+E58J/9DRov/gwi/wDiqP8AhOfCf/Q0aL/4MIv/AIqgDdorC/4Tnwn/ANDRov8A4MIv/iqP+E58J/8AQ0aL/wCDCL/4qgDdorC/4Tnwn/0NGi/+DCL/AOKo/wCE58J/9DRov/gwi/8AiqAN2isL/hOfCf8A0NGi/wDgwi/+Ko/4Tnwn/wBDRov/AIMIv/iqAN2isL/hOfCf/Q0aL/4MIv8A4qj/AITnwn/0NGi/+DCL/wCKoA3aKwv+E58J/wDQ0aL/AODCL/4qkPjrwkASfFOigDqTqEX/AMVQBvUUUUAFFFFABRRRQAUUUUAFFFFABWXfeItN03UIrO9kmiklKKJPsspiBdtqhpQuxSTwAWHJHqK1K5TxIuraj4gsdP8A7Gu7jQ42jubie3lgHnSq+UjYPIrBFKh2wCW4A4yCAaNp4u0e90+6voZp0s7RGkluZ7OaGPauclWdAGxg/dJpP+Ew0X+zvtnn3Gz7R9l8n7HN5/m7d+zydnmbtvzY2/d+bpzXJt4Z1WWy1Gz0fTr7TbA2mUs9RvlmWa5WVZF8vEknloQrKeVHzj5eDVn+y9a/t3/hJ/7GuN/9peb/AGZ50Pn+V9k8nfnf5e7dzjf93vnigDo5vGOhQ/Yi19uS9jSaGSOJ3QRuQqu7KpEaknALkDOfQ1t15h/wiWu2uj3enrYee+t6TFZTSxzIFsJPOndi24gsoFxxsBOU6DIr0a/tZry1MVvf3Fg+QfOt1jLD2/eKy/pQBl+Bv+SeeHf+wXbf+ilrdrC8DcfDvw5zn/iVWvP/AGyWt2gAor57/wCFn+L/APoL/wDktD/8RR/ws/xf/wBBf/yWh/8AiK7/AKjV7r+vkef/AGhS7P8Ar5n0JRXz3/ws/wAX/wDQX/8AJaH/AOIo/wCFn+L/APoL/wDktD/8RR9Rq91/XyD+0KXZ/wBfM+hKK+e/+Fn+L/8AoL/+S0P/AMRR/wALP8X/APQX/wDJaH/4ij6jV7r+vkH9oUuz/r5n0JRXz3/ws/xf/wBBf/yWh/8AiKP+Fn+L/wDoL/8AktD/APEUfUavdf18g/tCl2f9fM+hKK+e/wDhZ/i//oL/APktD/8AEUf8LP8AF/8A0F//ACWh/wDiKPqNXuv6+Qf2hS7P+vmfQlFfPf8Aws/xf/0F/wDyWh/+Io/4Wf4v/wCgv/5LQ/8AxFH1Gr3X9fIP7Qpdn/XzPoSivnv/AIWf4v8A+gv/AOS0P/xFH/Cz/F//AEF//JaH/wCIo+o1e6/r5B/aFLs/6+Z9CUV89/8ACz/F/wD0F/8AyWh/+Io/4Wf4v/6C/wD5LQ//ABFH1Gr3X9fIP7Qpdn/XzPoSivHPAnjvxJrPjWwsNS1HzrabzN8fkRrnEbMOQoPUCvY65atKVKXLI6qNaNaPNEKKKKyNgooryP8A4TfxD/0EP/IMf/xNcWKxlPC2509ex2YbB1MTfka07nrlFeR/8Jv4h/6CH/kGP/4mj/hN/EP/AEEP/IMf/wATXH/bOH7P8P8AM7P7Ir91+P8AkeuUV5H/AMJv4h/6CH/kGP8A+Jo/4TfxD/0EP/IMf/xNH9s4fs/w/wAw/siv3X4/5HrlFeR/8Jv4h/6CH/kGP/4mj/hN/EP/AEEP/IMf/wATR/bOH7P8P8w/siv3X4/5HrlFeR/8Jv4h/wCgh/5Bj/8AiaP+E38Q/wDQQ/8AIMf/AMTR/bOH7P8AD/MP7Ir91+P+R65RXkf/AAm/iH/oIf8AkGP/AOJo/wCE38Q/9BD/AMgx/wDxNH9s4fs/w/zD+yK/dfj/AJHrlFeR/wDCb+If+gh/5Bj/APiaP+E38Q/9BD/yDH/8TR/bOH7P8P8AMP7Ir91+P+R65RXkf/Cb+If+gh/5Bj/+Jo/4TfxD/wBBD/yDH/8AE0f2zh+z/D/MP7Ir91+P+R65RXkf/Cb+If8AoIf+QY//AImtfwt4p1nUvEtra3t55kEm/cvlIM4RiOQM9QKunm1CpNQSd27dP8yKmV1qcHNtaa9f8j0WsLxz/wAk88Rf9gu5/wDRTVu1heOf+SeeIv8AsF3P/opq9Y8o3aKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDC8Df8k88O/8AYLtv/RS1u1heBv8Aknnh3/sF23/opa3aAPk+iiivpz5UKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOs+GH/JR9L/AO2v/ol6+hK+e/hh/wAlH0v/ALa/+iXr6Erx8d/EXp/me1l/8J+v+QUUUVwHoBXgte9V4LXzud/8u/n+h7+Tfb+X6hRRRXzx9AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFb3gj/AJHKx/7af+i2rBre8Ef8jlY/9tP/AEW1dGF/3iHqvzOfFfwJ+j/I9crC8c/8k88Rf9gu5/8ARTVu1heOf+SeeIv+wXc/+imr7s+IN2iiigAooooAKKKKACiiigAooooAKKKpT61pVrqcWnXOpWcN9OAYrWSdVlkB/uoTk9D0HagC7RVGDW9KuWu1t9Ts5WsSRdCO4Vjb4znfg/L0PXHQ1GfEeiLpA1U6zp404ttF4bpPJJzjG/OM5460AaVFUptZ0u2uLWC41K0imvf+PWOSdVaf/cBOW6jp61doAwvA3/JPPDv/AGC7b/0UtbtYXgb/AJJ54d/7Bdt/6KWt2gD5Pooor6c+VCiiigAooooAKKKKACiiigAooooAKKKKACiiigDrPhh/yUfS/wDtr/6JevoSvnv4Yf8AJR9L/wC2v/ol6+hK8fHfxF6f5ntZf/Cfr/kFFFFcB6AV4LXvVeC187nf/Lv5/oe/k32/l+oUUUV88fQBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABW94I/wCRysf+2n/otqwa3vBH/I5WP/bT/wBFtXRhf94h6r8znxX8Cfo/yPXKwvHP/JPPEX/YLuf/AEU1btYXjn/knniL/sF3P/opq+7PiDdooooAKKKKACiiigAooooAKKKKACuN8WXYvddstBexvktXkhu7u+i06aZG8uUNHErohAYsoLMSNq+7AjsqKAPIk0fUrnw/YWkWlXizaPon2TU1kt2T7ZKJ7d2VCwAl3LDOcrkHzRzljWqqzJ4jPiN9N1A6S2rvMsQsZTMM2SQ+d5G3zAN6uv3c/Nu6EmvSKKAPI49G1Oz0OTTrjS7w3WpaRZW1iY4GdbV0mlbY7qCsfliSNskgHbxkivU79L6S1K6XcW9vcZGJLi3aZcd/lV0P61ZooAwvA2f+Fd+HM9f7Ktf/AEUtbtYXgb/knnh3/sF23/opa3aAPk+iiivpz5UKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOs+GH/JR9L/7a/8Aol6+hK+e/hh/yUfS/wDtr/6JevoSvHx38Ren+Z7WX/wn6/5BRRRXAegFeC171XgtfO53/wAu/n+h7+Tfb+X6hRRRXzx9AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFb3gj/kcrH/ALaf+i2rBre8Ef8AI5WP/bT/ANFtXRhf94h6r8znxX8Cfo/yPXKwvHP/ACTzxF/2C7n/ANFNW7WF45/5J54i/wCwXc/+imr7s+IN2iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAwvA3/ACTzw7/2C7b/ANFLW7WF4G/5J54d/wCwXbf+ilrdoA+T6KKK+nPlQooooAKKKKACiiigAooooAKKKKACiiigAooooA6z4Yf8lH0v/tr/AOiXr6Er57+GH/JR9L/7a/8Aol6+hK8fHfxF6f5ntZf/AAn6/wCQUUUVwHoBXgte9V4LXzud/wDLv5/oe/k32/l+oUUUV88fQBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABW94I/5HKx/7af+i2rBre8Ef8jlY/8AbT/0W1dGF/3iHqvzOfFfwJ+j/I9crC8c/wDJPPEX/YLuf/RTVu1heOf+SeeIv+wXc/8Aopq+7PiDdooooAKKKKACiiigAooooAKKKKACiiuP1a2uh48eKxv7uN7/AEO8PlvdyeSkiPAqOqZ2oRvb5gM8mgDsKK80juLuxsU8O3sl3a3a31jFf3Uerz3W+OXOCkz4eMs0e0gBSN4IPINVje3V1FNZXWs31vZ6fbatLa3SXjo8hgnCIzuDmTywSMMSG6sGNAHqlFeaQapqVzff2pdXd1DqEWrafZJYrO6xGGWCBpQYs7WOZZm3EZHlgZ+U16Ff3U1namW3sLi/fIHk27Rhj7/vGVf1oAy/A3/JPPDv/YLtv/RS1u1heBufh34c4x/xKrXj/tktbtAHyfRRRX058qFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHWfDD/ko+l/8AbX/0S9fQlfPfww/5KPpf/bX/ANEvX0JXj47+IvT/ADPay/8AhP1/yCiiiuA9AK8Fr3qvBa+dzv8A5d/P9D38m+38v1Ciiivnj6AKKKKACiiigAooooAKKKKACiiigAooooAK3vBH/I5WP/bT/wBFtWDW94I/5HKx/wC2n/otq6ML/vEPVfmc+K/gT9H+R65WF45/5J54i/7Bdz/6Kat2sLxz/wAk88Rf9gu5/wDRTV92fEG7RRRQAUUUUAFFFFABRRRQAVyHjq0uJLjRLizv9WiuUvCltbaf9mAkkMMuWZpkIXCb/UdsZII6+uX8f29hc6BAmqXGk28IulZW1Wya6iLBWwAqyIQ3cHPQHjnIALXhGxu7Kwu31KK8S7ubkyyveTxSPKfLRA37oKgG1AMAD7ue9WrrwxoF9qR1G90PTbi9K7Tcy2kbyEYxjcRnGOPpWL8OQF0W/SKOwECXzCGbTrKS1huF8uM7wru5PJKkg4yuB0yeuoAzofDui2+mS6bb6PYRWMxzLapaosUh/wBpAMHoOo7Us3h/RriztrS40ixltrQhreF7ZGSEjoUUjC49q0KKAKj6Tp0mqR6nJYWr38abEu2hUyovPAfGQOTxnvVuiigDC8Df8k88O/8AYLtv/RS1u1heBv8Aknnh3/sF23/opa3aAPk+iiivpz5UKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOs+GH/JR9L/AO2v/ol6+hK+e/hh/wAlH0v/ALa/+iXr6Erx8d/EXp/me1l/8J+v+QUUUVwHoBXgte9V4LXzud/8u/n+h7+Tfb+X6hRRRXzx9AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFb3gj/AJHKx/7af+i2rBre8Ef8jlY/9tP/AEW1dGF/3iHqvzOfFfwJ+j/I9crC8c/8k88Rf9gu5/8ARTVu1heOf+SeeIv+wXc/+imr7s+IN2iiigAooooAKKKKACiiigArm/Gr3P2PTYLU3Ti4vhHLbWVx5E9ygikYpHJldpyoY/MuQjDPPPSVR1bR7bWbaOK6M0bQyCWGaCVo5InAI3Kw5HDMPQgkHINAGT4Ke5+x6lBdG6QW98Y4ra9uPPntkMUbBJJMtuOWLD5mwHUbjjjpKo6To9to1tJFamaRppDLNNPK0kkrkAbmY8k4VR6AAAYAq9QAUUUUAFFFFAGF4G/5J54d/wCwXbf+ilrdrC8Df8k88O/9gu2/9FLW7QB8n0UUV9OfKhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB1nww/5KPpf/bX/wBEvX0JXz38MP8Ako+l/wDbX/0S9fQlePjv4i9P8z2sv/hP1/yCiiiuA9AK8Fr3qvBa+dzv/l38/wBD38m+38v1Ciiivnj6AKKKKACiiigAooooAKKKKACiiigAooooAK3vBH/I5WP/AG0/9FtWDW94I/5HKx/7af8Aotq6ML/vEPVfmc+K/gT9H+R65WF45/5J54i/7Bdz/wCimrdrC8c/8k88Rf8AYLuf/RTV92fEG7RRRQAUUUUAFFFFABRRRQAUUUUAFcV4n1zVPDuu6hdRXj3NpH4fvdQjsHjQIkkHk7cMF3nO9s5YjngDFdrWPd+FNJvtcGr3cd1JdrE0ODfTiIxtjchiD+WVO0ZBXBIBPNAHLz6rrmn6wfDj61Jcy3ElkV1IwQiSFZvtG9VUJs/5djtypx5gzuxVU+Iteu7a5todWNtPpdtqEz3IgjP2tre4MUe8FcBcKS4TacngrXWxeDNDh02axS1m8qZ0kZ2u5mlDJjYVlLb124G3awx2xRceCtAurG2s5rJjBbJIiBbiVSyyEGRXYMDIrkAsHJDHk5oA56HxLq11qH9qLdtFZpqtrp503y0KsksMTFy23fvDTZ4IG1enOa7W/v4dNtTcXCXDoCBi3tpJ25/2Y1ZvxxVN/DWkvri6u1s32xWDgiZxGXC7A5jzsLhTtDEZA4zWrQBheBjn4d+HD/1CrX/0UtbtYXgb/knnh3/sF23/AKKWt2gD5Pooor6c+VCiiigAooooAKKKKACiiigAooooAKKKKACiiigDrPhh/wAlH0v/ALa/+iXr6Er57+GH/JR9L/7a/wDol6+hK8fHfxF6f5ntZf8Awn6/5BRRRXAegFeC171XgtfO53/y7+f6Hv5N9v5fqFFFFfPH0AUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVveCP+Rysf+2n/AKLasGt7wR/yOVj/ANtP/RbV0YX/AHiHqvzOfFfwJ+j/ACPXKwvHP/JPPEX/AGC7n/0U1btYXjn/AJJ54i/7Bdz/AOimr7s+IN2iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAwvA3/JPPDv/AGC7b/0UtbtYXgb/AJJ54d/7Bdt/6KWt2gD5Pooor6c+VCiiigAooooAKKKKACiiigAooooAKKKKACiiigDrPhh/yUfS/wDtr/6JevoSvnv4Yf8AJR9L/wC2v/ol6+hK8fHfxF6f5ntZf/Cfr/kFFFFcB6AV4LXvVeC187nf/Lv5/oe/k32/l+oUUUV88fQBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABW94I/wCRysf+2n/otqwa3vBH/I5WP/bT/wBFtXRhf94h6r8znxX8Cfo/yPXKwvHP/JPPEX/YLuf/AEU1btYXjn/knniL/sF3P/opq+7PiDdooooAKKKKACiiigAooooAKKKKACiiigAoriYfiC0y6q4tNPf7DdvZR20Wpb7l5vtP2ePzI/LHlq7EHdubAI4NWR4xv5Zhptvo8D64tzLBNatelYUCRpKXEvl5IKyxY+Qcvg4wTQB1tFcYvxBFxDBe2OmmTTRBaT3k8k+ySAXDYUBNpDlRy3zLgHjceK6q/wBSsdKtTc6peW9lbghTLcSrGoJ6DLECgDL8Df8AJPPDv/YLtv8A0UtbtYXgY5+Hfhwjkf2Va/8Aopa3aAPk+iiivpz5UKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOs+GH/ACUfS/8Atr/6JevoSvnv4Yf8lH0v/tr/AOiXr6Erx8d/EXp/me1l/wDCfr/kFFFFcB6AV4LXvVeC187nf/Lv5/oe/k32/l+oUUUV88fQBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABW94I/5HKx/7af8Aotqwa3vBH/I5WP8A20/9FtXRhf8AeIeq/M58V/An6P8AI9crC8c/8k88Rf8AYLuf/RTVu1heOf8AknniL/sF3P8A6Kavuz4g3aKKKACiiigAooooAKKKKACiiigAooooA5HWPBd7r98lzqWrW6m0LPYfZrExtE/mpKjSEyN5mGiQ4GwHk+mFHg6/imGpW+sQJrjXMs8101kWhcSRpEUEXmZACxRY+c8pk5yRXW0UAcYvw+FvDBZWOpGLTTBaQXkMkG+ScW7ZUh9wCFhgN8rZA42nmuzoooAwvA3/ACTzw7/2C7b/ANFLW7WF4G/5J54d/wCwXbf+ilrdoA+T6KKK+nPlQooooAKKKKACiiigAooooAKKKKACiiigAooooA6z4Yf8lH0v/tr/AOiXr6Er57+GH/JR9L/7a/8Aol6+hK8fHfxF6f5ntZf/AAn6/wCQUUUVwHoBXgte9V4LXzud/wDLv5/oe/k32/l+oUUUV88fQBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABW94I/5HKx/7af+i2rBre8Ef8jlY/8AbT/0W1dGF/3iHqvzOfFfwJ+j/I9crC8c/wDJPPEX/YLuf/RTVu1heOf+SeeIv+wXc/8Aopq+7PiDdooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAMLwN/yTzw7/2C7b/0UtbtYXgb/knnh3/sF23/AKKWt2gD5Pooor6c+VCiiigAooooAKKKKACiiigAooooAKKKKACiiigDrPhh/wAlH0v/ALa/+iXr6Er57+GH/JR9L/7a/wDol6+hK8fHfxF6f5ntZf8Awn6/5BRRRXAegFeC171XgtfO53/y7+f6Hv5N9v5fqFFFFfPH0AUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVveCP+Rysf+2n/AKLasGt7wR/yOVj/ANtP/RbV0YX/AHiHqvzOfFfwJ+j/ACPXKwvHP/JPPEX/AGC7n/0U1btYXjn/AJJ54i/7Bdz/AOimr7s+IN2iiigAooooAKKKKACiiigAooooAKKKKACiiigDF17xPb6DJHHJZ3l7K0Mly8doisY4Y9u+RtzLwNy8DLHPAODWvDLHcQJNC4eORQyMvRgRkGuc8TaJqt5qC3uhNZmWSwnsJVu5GRUWQqVkXarbipU/Kcbs/eGOdUaJbHQrbSnkuRBbxJGrQXMkDkIAB80bKe3TNAFPwN/yTzw7/wBgu2/9FLW7XPW/gfR7S1itrWXV4YIUEccaa3eBUUDAAHm8ACpP+EP03/n51r/weXv/AMdoA+bKK99/4VJ4L/6BU3/gwuf/AI5R/wAKk8F/9Aqb/wAGFz/8cr1vr8ex4/8AZ8v5jwKivff+FSeC/wDoFTf+DC5/+OUf8Kk8F/8AQKm/8GFz/wDHKPr8ewf2fL+Y8Cor33/hUngv/oFTf+DC5/8AjlH/AAqTwX/0Cpv/AAYXP/xyj6/HsH9ny/mPAqK99/4VJ4L/AOgVN/4MLn/45R/wqTwX/wBAqb/wYXP/AMco+vx7B/Z8v5jwKivcIvhF4ZGvXTy6bKdPNtCIE/tC4yJQ0vmH7+eVMX5fWrv/AAqTwX/0Cpv/AAYXP/xyj6/HsH9ny/mPAqK99/4VJ4L/AOgVN/4MLn/45R/wqTwX/wBAqb/wYXP/AMco+vx7B/Z8v5jwKivff+FSeC/+gVN/4MLn/wCOUf8ACpPBf/QKm/8ABhc//HKPr8ewf2fL+Y8Cor33/hUngv8A6BU3/gwuf/jlH/CpPBf/AECpv/Bhc/8Axyj6/HsH9ny/mPLvhh/yUfS/+2v/AKJevoSuStfhf4Usblbiysru3nTO2WLU7pGXIwcESZHBIrQ/4Q/Tf+fnWv8AweXv/wAdrhxFZVp8yR34ai6MOVs3aKwv+EP03/n51r/weXv/AMdo/wCEP03/AJ+da/8AB5e//Ha5zpN2vBa9d/4Q/Tf+fnWv/B5e/wDx2qH/AArHwr/z53n/AINLr/45XmY/BSxXLyu1r/jY9LA4yOF5rq97fqeY0V6d/wAKx8K/8+d5/wCDS6/+OUf8Kx8K/wDPnef+DS6/+OV5n9i1P50ej/bFP+VnmNFenf8ACsfCv/Pnef8Ag0uv/jlH/CsfCv8Az53n/g0uv/jlH9i1P50H9sU/5WeY0V6d/wAKx8K/8+d5/wCDS6/+OUf8Kx8K/wDPnef+DS6/+OUf2LU/nQf2xT/lZ5jRXp3/AArHwr/z53n/AINLr/45VKL4X6INeunlt7s6ebaEQJ/alzkShpfMP+szypi/L60f2LU/nQf2xT/lZ59RXp3/AArHwr/z53n/AINLr/45R/wrHwr/AM+d5/4NLr/45R/YtT+dB/bFP+VnmNFenf8ACsfCv/Pnef8Ag0uv/jlH/CsfCv8Az53n/g0uv/jlH9i1P50H9sU/5WeY0V6d/wAKx8K/8+d5/wCDS6/+OUf8Kx8K/wDPnef+DS6/+OUf2LU/nQf2xT/lZ5jW94I/5HKx/wC2n/otq7D/AIVj4V/587z/AMGl1/8AHKkg+HPhu1mWa2h1CGVfuvHq12rDt1Eta0cpnTqRm5LRpmdXNYVKcoKL1TR1FYXjn/knniL/ALBdz/6Kaj/hD9N/5+da/wDB5e//AB2o7jwPo93ay211Jq80EyGOSN9bvCrqRggjzeQRX0J4B0NFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAeU+I/Get6dqXiEaVq/2jULG/8AIstE+zI/mQ/ZI5GfAXzG2szNkNj5dp6ip7rxTfR2V0bHxbb3WnW91CBqks1rA84aJy8Mchj8kurCNuV6FlyCM16Na6fbWU13LbRbHvJvPnO4ne+xUzyePlRRgccVZoA8n1HV5b3ULO9fV7rRrJb2xlkupIoIZESSzkOZiyFc7iow3AJwO1W7nxRqujWclxAySJe6hqOm2sSwIo+2GdhbE7V6Ha4YnqcE89fTaKAPPLTXdfPj46bc6xp8UdvdLB9iurhEmuofKBMqxCHczEktuWQKMEbeDXodFFABRRRQAUUUUAFFFNllSCF5ZWCxxqWZj2A5JoAdRUEF7BdCJrdmkSaITRyKjbGQ4wd2MZ56ZzU9ABRRUMt5bw3cFtLKqzXG7ykPV9oycfQUATUUUUAFFFFAGJ4zv7vSvA2t3+myiG7trGaWCQqG2OqEg4PB57GuF1zxJ4j0nxPPof8Ab0MVpGDKdU1CWG12MUjKQl/IdOcyMAUDEL97jn1WigDzC88WaxFfQJda9BbXv9oaVbpYW0SNHeQTNB50yF137S0kqBgQAE6bjmsnWfFurarF4j0+21KTyJdI1GaOJJYnuLR4JEUKVWFfLyGb5WZ2xggqRk+qXWg2F9qkN/dpPLLAyvGjXUvkhl+63lbthYdQSuenoK0aAPOp/EE1hqGoajY6wmo6fp9rp9zcXYjgY3Fu0tyszM8aAEKo3jGMeXx95s52oeK/FCw2Mh1Ww0hb2ya/t5dSnS3RmklYpAcwvv2R+UCqlHO4nd6erUUAIudo3YDY5xS0UUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABWb4isv7S8N6hZGO4lFxA0ZS1dFlYEYwpchc/wC9x68VpUUAePw+B9YnvhJqOkXVtaKipZR6VHp8L2zLI7GV1bcI3bcnzQsSdnIHAGy3gW4abzxpkK3F3qmpfbJmKEvaTR3ARW55Qs0J2c4PJHWvR6KAPK28F6hPoNpZ23h/+z0XS4rWaAPCm6UXMLufkcjlUZs5yfrxWjH4HmsdYkutJ02GzaPW/MtZotg8i0a1CuFGflQys5KDqTux3r0OigDyXT/AOoJoT2lxYaibuR7Bb0zS2awXPl3UTyyKYgrudqud0nzkHHJNai+C2tPEdrcHw5BdadZ6rO9rAiwYt4pIYsSIrMAqiVXJA+bPIBr0aigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigD/9k=)

**#Data Processing**

**def preprocess\_text(sen):**

**# Removing html tags**

**sentence = remove\_tags(sen)**

**# Remove punctuations and numbers**

**sentence = re.sub('[^a-zA-Z]', ' ', sentence)**

**# Single character removal**

**sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)**

**# Removing multiple spaces**

**sentence = re.sub(r'\s+', ' ', sentence)**

**return sentence**

**TAG\_RE = re.compile(r'<[^>]+>')**

**def remove\_tags(text):**

**return TAG\_RE.sub('', text)**

**#Next, we will pre-process our reviews and will store them in a new list**

**X = []**

**sentences = list(movie\_reviews['review'])**

**for sen in sentences:**

**X.append(preprocess\_text(sen))**

**#we will convert labels into digits**

**y = movie\_reviews['sentiment']**

**y = np.array(list(map(lambda x: 1 if x=="positive" else 0, y)))**

**#dividing our dataset into train and test sets**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)**

**#preparing the embedding layer**

**tokenizer = Tokenizer(num\_words=5000)**

**tokenizer.fit\_on\_texts(X\_train)**

**X\_train = tokenizer.texts\_to\_sequences(X\_train)**

**X\_test = tokenizer.texts\_to\_sequences(X\_test)**

**vocab\_size = len(tokenizer.word\_index) + 1**

**maxlen = 100**

**Xtrain = pad\_sequences(X\_train, padding='post', maxlen=maxlen)**

**X\_test = pad\_sequences(X\_test, padding='post', maxlen=maxlen)**

**#GloVe embeddings to create our feature matrix**

**from numpy import array**

**from numpy import asarray**

**from numpy import zeros**

**embeddings\_dictionary = dict()**

**glove\_file = open('E:/Datasets/Word Embeddings/glove.6B.100d.txt', encoding="utf8")**

**for line in glove\_file:**

**records = line.split()**

**word = records[0]**

**vector\_dimensions = asarray(records[1:], dtype='float32')**

**embeddings\_dictionary [word] = vector\_dimensions**

**glove\_file.close()**

**embedding\_matrix = zeros((vocab\_size, 100))**

**for word, index in tokenizer.word\_index.items():**

**embedding\_vector = embeddings\_dictionary.get(word)**

**if embedding\_vector is not None:**

**embedding\_matrix[index] = embedding\_vector**

**#next, text classification with simple neural network**

**model = Sequential()**

**embedding\_layer = Embedding(vocab\_size, 100, weights=[embedding\_matrix], input\_length=maxlen , trainable=False)**

**model.add(embedding\_layer)**

**model.add(Flatten())**

**model.add(Dense(1, activation='sigmoid'))**

**model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['acc'])**

**print(model.summary())**

![A screenshot of a cell phone

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDcRXhpZgAATU0AKgAAAAgABAE7AAIAAAAGAAAISodpAAQAAAABAAAIUJydAAEAAAAMAAAQyOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEhhcnNoAAAFkAMAAgAAABQAABCekAQAAgAAABQAABCykpEAAgAAAAM5MQAAkpIAAgAAAAM5MQAA6hwABwAACAwAAAiSAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAxOToxMDowNyAxMjozMjoyOQAyMDE5OjEwOjA3IDEyOjMyOjI5AAAASABhAHIAcwBoAAAA/+ELGGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTAtMDdUMTI6MzI6MjkuOTEwPC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkhhcnNoPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIAO4CqQMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APoePPmDbyecZPt+Pr+vvzi+Htd13VdZ1Kz1TR9OsodOlEEktvqTzs0hjjkGFaBPl2yDnOcjoetbUZ/eAgbupwO/H+fz6888NFqHim0bxPNZeDNZjudTm86ykM9gRGwto4lLg3H9+MnoeMfShbv0/wAhpXOt1XxNp2jXCw363wJXeZIdOuJo0X1aREKoODncRgc0l34o0myvtLtZZ5Xk1c4smgtpZkl4zneilVGOcsQMZPQGuU1nS9fvdWuxeabrN/5iRixlstb+w2sA8sbhMI5Fctv3EnZIMFQMcipLzTrmL4N2L38LaRqmgWcVzEbl0fyZrdOpKMwKMFZTgklXPAPFGiV30t93X7gs3ZLr+Z0h8XaMuo6nZSXE0U2lRLLeGW1lRI1b7pDlQrZwcBSScHHSpNK8TaZrNwbe0a6inClxDe2M9o7KMAsqyopYAkAkZAyK5a28K6jq/wAPbqW9Q2+uaxcx6pcRfaXi2uro8duZUAZAqIke5eQQSKs+HtF265bXb+H9esntg+ZtX8QSXSKSpH7uPz5QxOcZYJgc9eKdrOzJvdXRpSePvD0NxdW8tzdLPZvtuITp9xviH/PQrsz5fP8ArPuf7VdGrB1DKQykZBB4IrjtHu9Xg8YatPceEtVit9QnhCXLTWZVFRAhZgJy2M5OACcds8V2VJbD6iMwRSzEBQMknsK47wr42uNa8I33ifULQpprXMg0+O0t5ZppLdW2B2RQzFiwY/KMBcH1NdZeWwvLC4tWYqJomjLDqMjGf1riPhLLIfhyug3Ufk6jokkum3UZUgB1JKkcchkZWyPWtVH9xOS3Vvu1v+Nl8w6xvtf+v1+42/DPjOy8TWFtNBZ6lbyy2i3LrPptzHGuQCVWV41V+vG089QMUL488Pt4Vm8Rie7GlwyGOSY6dcBlIOD+7Me/APBbGAQeeDVXwnca9aafpWi3vhyW0isbVLe5vZryExuUTaDEqFmYEgffEeAc9RisOfRLif4ly+H4CjeH5JI9fvIs8xzbioixjG2SVFl69Y39ahpc1lt/X6X/AKYbK/8AX9XOik+IPh+K/ksWk1Br2JVd7RNJu3mCMOH8sRFivGN2MA8E54rQu/FGkWWm2189y88N0dsAtLeS4klPfbHGrMcYOeOMc4rnnvtQ034la3NY6HNq0UljZLILSWJJoyDORnzXRSvJ/iyD2OeKV94Ov1jsNQktry8kjmvJLiw03VJLKRRcyiXEbq8YcqQAQzKrZLZyADPQeh1y6x/bGg3lx4aeN72ON1ijvIZIvLm25VZY2CuvVcggHBzVbwR4nXxf4PsdYMP2eaVSlxB/zylRirr/AN9A49sVR0OO18NaVq2s3um6jpdusQmmbU9UkvrgrGrE5zJIFABOArnOTwOM1PhFp11ZeAI7rUIWt7jVbqfUWgb/AJZCVyyrjt8u04963jGLpSfmv10/r9SLvT5/1+X3ncUUUVgUFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFc7feIdTk1m503w3o8WovZKpu5rq8NtEjMNwjUiNyz7SGIwAAy5bnFdFXKy/214e17U57DRpdZsNTkW4UWs0UctvMI1jYMJXUFCI0IIJIJYbSMUdR9C5pfjDTtS+wRsl1a3V9JNCtvNbufLli3eZG7qCisNrEZb5gMrkVIPFemPYajdQrfSDTZPKuIhp1x5ofAIxHs3sCGB3KCMHOcVgjRtb0yz0zU5LH+0tQj1Wa/vbSylRcCWKSMLG0hRW2B0GSV3bSeOlW/DMeuS6t4il1rRjpq38sctswuY5VK+Ske1ipyHBUkjG0Z4ZutKV+V23t+IK1/L9C14b8a2PiPTYLmKz1O3kks1unSXS7pEUFQSqSNEqyHnjbkt1Apll8RPDWoFfsl5cvGXMbzNp9wsULg4KSyGMLEwIxtcqeRxyKh8J3GvWmn6Vot74cltIrG1S3ub2a8hMblE2gxKhZmBIH3xHgHPUYrA0T+17/wAEX/h+HQGaHUJtQhj1JJYvs8ayTygtIhcSbhk8KrBuORk7bklzaArbM7hvEOni41KAfamm0xEe5jSymZtrglSgCZkztb7m7kEday9Y8eabpeh6ZqsMN7e22pSwrC1vY3D/ACO6ruIWMlSA2QrAFiMDmortda0bxRdXWl6LLq8d7Y28EcouoolikiaXPmljuCkSA5RXPB+XpmjZaXrY+FukWc2ksmqabLayS2ZnjzL5MyOwRgxX5gp27iOozt7T/mv+CC8zrNK1e21m1a4s47yNFfYReWU1q2cA8LKisRz1Ax+VXqy9H1DU9Rad9R0STSYV2iFbi4jkmfruLLGWVR0xh2JycgY51KbEcdL4y1iWG5vtG8PRalp1vdPaN5V8wu96SGNiYPKIADAn7+7bzt7U+58bQ6L4v1LS9duFaJLeC5tI7KwnlmVG8wOZAm8lQUB37VA3YPrWLd6Pqev+W+o+Co9N8RGRSuv2stuFhCuCDvD+dnaOU2lSTgkjmugt/wC0Lfx1rt62g3jWz2ECQXCyW+LhojKxRQZdwJ80AbgoyGyRwTK2Hpdmtd6/YWelwagWuLm1uApiextJbouCMghYlY4xznGKqz+MtDtvD8utTXUq2ED7J3FrKWt2HUSxhd8eOp3AY74rkodO8Qr4T8O2l1outC2tYHivtP07UYba5Mg2iN/NWZQY+H4WQHJXIPIFWfTNT074beOLXUdKnsxcB7i3a7v/ALasiNAibWkZjISCh3BhjnCsw5qurFskd1pfizR9Zuhb2M0+9wWiM9pNCk4HUxPIoWUY5yhPGD0qOLxnok2ofZEnuATL5K3DWM62zvnbtWcp5THdwAGOTx1qkV1bxDqWmi/0W40dNNuDcSTyTwyLK3lSRhYSjlsfvM5dUOBjGTxzOn+DZ7Oxh0W80HXr1oFEQul8UTrYyqvCsyfaN6ZAyUETAHgZHNGlwlotD1CiiikAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFZfibVp9B8L6jq1rax3cljbvceRJMYg6oNzDcFbBwDjjk4HHWtSuf8AHlvf3vgLWbLSLKW+vLyzkt4oYnjUkupXJMjKuBnJ5zjoCeKT2KhbmV9hq69q6+J9J0q40qySO+spbqaeO/djC0ZQMqqYRvGZUwSV43cDABsr4s0k6qNPka8gmaXyUe4sJ4YZHzgKszoI2JPQBjntmuZ03w5Do/jfR77QPA66Rbvps0N5PbRWkQjkkaJlWQJJubb5bZ2hh8wwTk4x5dF8SyQ2M9zoGvXmqWt7b3N7NLr6fZ7jZMrN9ng87y8cZAdIsDvu4N6XS/rf/IlXcb9TvdU8W6Po2qR6fqU80NzNGXhH2SZlm/2UcKVd/wDYUluelK/izR49Fi1RriX7PNKYY0FrKZmkBOYxCF8zeNrZXbkYPHFZPiC41WPxjpV3Z+F9S1G2sYpt81vNaKCZFUAKJJlORg5yB7Zq74jil1LTbISeHru+Vj5zpBepb3No2w42OHX5/mK5WQDBPzEHmNbB1K2s+MEbwhrGp+HbiNLrSofOlj1CymTZhd+1kYxspK9+2QSD0rS0fxZo2vXTW+mXMjyiPzUEttLCJY843xl1AkTJHzJkcjnkVzF3p2vS/D/xFZLpWoTLeRmDTtOuryKe5jVowreZK8pUjcWODIxA7nhRsXLalceM9Auk0G9S1SzmW4laS3xbNJ5ZCMBLkkeWc7Aw5GCecVpcOh1Fcx4w8YQeGo4Y0u7FLqQSMYriQBtoicqduQeXVV/EjrXT1xXjjSL/AFTULFbO5v1i2kMsEMbImXjQnJQnO12PXomccGsKzko+7uZVpSjTbjudha3MV5ax3FtNFPFIoZZImDIw9QR1FS1HbxPBbpHJPJcOowZZQoZ/c7QB+QFSVsaRvZXCiiigYUUUUAFFFFAFaH/Wj/P+ev6+/Nms+eYwW00ysitHGzhnBKjCk5OMnH0/rz4N/wAL71//AKGTwH/4A6r/APG66aGGqYi/IthN2V2fRNZur+HNE8QCIa9o+n6mIc+V9ttUm8vOM7dwOM4HT0rxvxF8ata0nXZ7OHX/AAXEkaxkJc2epNINyK3JRCvfjHbGeaL341a1b+H9Lvk1/wAFrJdtMHeSz1IxvsYAbAE3DGed3titv7PrtJ239f8AIXOj3C0s7bT7OK0sLeK1toVCRQwoERFHQBRwB9Kmrwmw+NWtXWharePr/gtns1iKPHZ6kI03PtO8FNxz229+vFHh341a1q3iC0sZ9f8ABc0czEFLWz1JZD8pPyl0C9u9N5fXs21t6+vYXMj3aivnb/hfev8A/QyeA/8AwB1X/wCN1p+IvjVrWk67PZw6/wCC4kjWMhLmz1JpBuRW5KIV78Y7YzzT/s3EXtb8/wDIOdHu1FeE3vxq1q38P6XfJr/gtZLtpg7yWepGN9jADYAm4Yzzu9sUWHxq1q60LVbx9f8ABbPZrEUeOz1IRpufad4Kbjntt79eKX9n17Xt5de9uwc6Pdqx9M8I+G9FvTeaP4e0rT7oqVM9rZRxPg9RuVQcGvIfDvxq1rVvEFpYz6/4LmjmYgpa2epLIflJ+UugXt3rM/4X3r//AEMngP8A8AdV/wDjdNZbiOa1vz/yDnVj3aHw1oVvrT6xBounRam5Je9S0RZmJGDmQDccj3rTrwnxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5ovfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YpLL67Sst9t/XsHOj3aivCbD41a1daFqt4+v+C2ezWIo8dnqQjTc+07wU3HPbb368UeHfjVrWreILSxn1/wXNHMxBS1s9SWQ/KT8pdAvbvQ8vrpN229f8g50e7UV87f8L71/wD6GTwH/wCAOq//AButPxF8ata0nXZ7OHX/AAXEkaxkJc2epNINyK3JRCvfjHbGeaf9m4i9rfn/AJBzo92orwm9+NWtW/h/S75Nf8FrJdtMHeSz1IxvsYAbAE3DGed3tiiw+NWtXWharePr/gtns1iKPHZ6kI03PtO8FNxz229+vFL+z69r28uve3YOdHu1FeE+HfjVrWreILSxn1/wXNHMxBS1s9SWQ/KT8pdAvbvWZ/wvvX/+hk8B/wDgDqv/AMbp/wBm4i9rfn/kHOj6JorwnxF8ata0nXZ7OHX/AAXEkaxkJc2epNINyK3JRCvfjHbGeaL341a1b+H9Lvk1/wAFrJdtMHeSz1IxvsYAbAE3DGed3tiksvrtJ239f8g50e7UV4TYfGrWrrQtVvH1/wAFs9msRR47PUhGm59p3gpuOe23v14o8O/GrWtW8QWljPr/AILmjmYgpa2epLIflJ+UugXt3oeX10m7bev+Qc6PdqK+dv8Ahfev/wDQyeA//AHVf/jdafiL41a1pOuz2cOv+C4kjWMhLmz1JpBuRW5KIV78Y7YzzT/s3EXtb8/8g50e7UV4Te/GrWrfw/pd8mv+C1ku2mDvJZ6kY32MANgCbhjPO72xRYfGrWrrQtVvH1/wWz2axFHjs9SEabn2neCm457be/Xil/Z9e17eXXvbsHOj3aivCfDvxq1rVvEFpYz6/wCC5o5mIKWtnqSyH5SflLoF7d6zP+F96/8A9DJ4D/8AAHVf/jdP+zcRe1vz/wAg50fRNFeE+IvjVrWk67PZw6/4LiSNYyEubPUmkG5FbkohXvxjtjPNF78atat/D+l3ya/4LWS7aYO8lnqRjfYwA2AJuGM87vbFJZfXaTtv6/5Bzo92orwmw+NWtXWharePr/gtns1iKPHZ6kI03PtO8FNxz229+vFHh341a1q3iC0sZ9f8FzRzMQUtbPUlkPyk/KXQL270PL66TdtvX/IOdHu1FfO3/C+9f/6GTwH/AOAOq/8AxutPxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5p/wBm4i9rfn/kHOj3aivCb341a1b+H9Lvk1/wWsl20wd5LPUjG+xgBsATcMZ53e2KLD41a1daFqt4+v8Agtns1iKPHZ6kI03PtO8FNxz229+vFL+z69r28uve3YOdHuksUc8LxTIskcilXRxkMDwQR3FUNI8PaNoCSpoWkWGmLMQZFsrZIQ5HQnaBnrXjXh341a1q3iC0sZ9f8FzRzMQUtbPUlkPyk/KXQL271mf8L71//oZPAf8A4A6r/wDG6ay3EXat+f8AkHOj6JorwnxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5ovfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YpLL67Sdt/X/ACDnR7tRXhNh8atautC1W8fX/BbPZrEUeOz1IRpufad4Kbjntt79eKPDvxq1rVvEFpYz6/4LmjmYgpa2epLIflJ+UugXt3oeX10m7bev+Qc6PdqK+dv+F96//wBDJ4D/APAHVf8A43Wn4i+NWtaTrs9nDr/guJI1jIS5s9SaQbkVuSiFe/GO2M80/wCzcRe1vz/yDnR7tVTU9I03W7M2ms6fa6hbFgxhu4VlTI6HawIzXil78atat/D+l3ya/wCC1ku2mDvJZ6kY32MANgCbhjPO72xRYfGrWrrQtVvH1/wWz2axFHjs9SEabn2neCm457be/Xip/s+va9vz727D50e3WNhZ6ZZR2em2sFnawjEcFvGI0QZzgKOByasV4T4d+NWtat4gtLGfX/Bc0czEFLWz1JZD8pPyl0C9u9Zn/C+9f/6GTwH/AOAOq/8Axuq/s7EOVrfn/kLmifRNFeE+IvjVrWk67PZw6/4LiSNYyEubPUmkG5FbkohXvxjtjPNF78atat/D+l3ya/4LWS7aYO8lnqRjfYwA2AJuGM87vbFJZfXaTtv6/wCQc6PdqK8JsPjVrV1oWq3j6/4LZ7NYijx2epCNNz7TvBTcc9tvfrxR4d+NWtat4gtLGfX/AAXNHMxBS1s9SWQ/KT8pdAvbvQ8vrpN229f8g50e7UV87f8AC+9f/wChk8B/+AOq/wDxutPxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5p/2biL2t+f+Qc6PdqK8JvfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YosPjVrV1oWq3j6/4LZ7NYijx2epCNNz7TvBTcc9tvfrxS/s+va9vLr3t2DnR7tRXhPh341a1q3iC0sZ9f8ABc0czEFLWz1JZD8pPyl0C9u9Zn/C+9f/AOhk8B/+AOq//G6f9m4i9rfn/kHOj6JorwnxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5ovfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YpLL67Sdt/X/IOdHu1FeE2Hxq1q60LVbx9f8Fs9msRR47PUhGm59p3gpuOe23v14o8O/GrWtW8QWljPr/guaOZiClrZ6ksh+Un5S6Be3eh5fXSbtt6/5Bzo92or52/4X3r/AP0MngP/AMAdV/8AjdafiL41a1pOuz2cOv8AguJI1jIS5s9SaQbkVuSiFe/GO2M80/7NxF7W/P8AyDnR7tRXhN78atat/D+l3ya/4LWS7aYO8lnqRjfYwA2AJuGM87vbFFh8atautC1W8fX/AAWz2axFHjs9SEabn2neCm457be/Xil/Z9e17eXXvbsHOj3aivCfDvxq1rVvEFpYz6/4LmjmYgpa2epLIflJ+UugXt3rM/4X3r//AEMngP8A8AdV/wDjdP8As3EXtb8/8g50fRNFeE+IvjVrWk67PZw6/wCC4kjWMhLmz1JpBuRW5KIV78Y7YzzRe/GrWrfw/pd8mv8AgtZLtpg7yWepGN9jADYAm4Yzzu9sUll9dpO2/r/kHOj3aivCbD41a1daFqt4+v8Agtns1iKPHZ6kI03PtO8FNxz229+vFHh341a1q3iC0sZ9f8FzRzMQUtbPUlkPyk/KXQL270PL66TdtvX/ACDnR7tRXzt/wvvX/wDoZPAf/gDqv/xutPxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5p/2biL2t+f+Qc6PdqK8JvfjVrVv4f0u+TX/AAWsl20wd5LPUjG+xgBsATcMZ53e2KLD41a1daFqt4+v+C2ezWIo8dnqQjTc+07wU3HPbb368Uv7Pr2vby697dg50e7UV4T4d+NWtat4gtLGfX/Bc0czEFLWz1JZD8pPyl0C9u9Zn/C+9f8A+hk8B/8AgDqv/wAbp/2biL2t+f8AkHOj6Joqtp1w13pdpcu0btNCkhaIEISVByoPOPTPNWa89pp2ZSd1dBRRRSGVof8AWj/P+ev6+/Nmq0P+tH+f89f19+cuLxnok2ofZEnuATL5K3DWM62zvnbtWcp5THdwAGOTx1oDzN2iuUn+JfhW2v57K5v54rmCb7PJG9hcKRLjIjGY+WYcqoyXH3c1Dqfjm3s7jQdShv7ZtC1N5YXDWspuDIqOwCAHO7cm0x+WWzkcHijpcbTR2NFY9r4q0a8026v0umihs/8Aj5W5gkglh4yN0cih1yOmRz2zTtK8TaZrNwbe0a6inClxDe2M9o7KMAsqyopYAkAkZAyKBGtRXFeJfiBaWdjL/Yl4n2mGYL5lzp8720+Gw0aTDbGX+jtggjae2zrXivTNIlktJbmRbvy9xMVjNdJb5+68vlAhF7/My5APPGQdB2d7G5RXI2/ip5fhzFq+oXLRXFxHJGLvTNOmuljcFwJREnmHaNu7kkdiea2Jtcs9I8LRatqd3JNapAjyXUdq7FgQP3hjRSVHOTxhR1wBR38hdvM1qKxtJ8WaNrd81pp107ziMyqstvJEJowcGSMuoEiZI+ZMjkc8iqlz8QPDloZTNeXBjifyxNHYzvHK+duyN1QrK+eNqFjweODQB0lFNjkEsSSKGCuoYBlKnB9QeQfY1S1ibUINPeTSktTKoJZ7lm2oACchVHznOOMr9aUnyq7HFczsi/RVDQb+XVPDem6hcKiy3dpFO6oCFDMgYgZzxk1fqpJxdmTFqSTQUVy3irxRd6Vr3h7Q9Gggm1DWLohvP3bYreMbpn45zjhe2TUV74p16PxPqGlWOkaO0Vmtuy3F7rMlv5vnFlRdot3w25CMZOcrjrgNwaipPr/w35j62OuorkW+I2k6ba2a+KEuNI1C4tjcNam0uJQig/OQ4jAIXqxwNoIJAFR6v46tbjwjqureFNQtZJNLQTym8tZNjx4JBXJTKsAdsgJU4OM4qfMNbHZUVi6X4v0XWL4WljcymV1LxedaywrcKOrRM6hZQOuULDBB6GmReM9Em1D7Ik9wCZfJW4axnW2d87dqzlPKY7uAAxyeOtAG7RWHrXivTNIlktJbmRbvy9xMVjNdJb5+68vlAhF7/My5APPGRT0nxjB/wh+n6prk8bT3ZZFXT7aSUzsrMMxxJvcjAzxuwOc4o6XHZ3sdRRXLeHvFZ1a7113mjubKxKPD9nsZo5kUqxMbxEs5kG3oFUn+7V7wzqi6np95drfz3qC7lCiawe1kt1ByIjGyhiVB6kZOaBG3RXNWfxC8N39xDFb3s4M0ogDy2M8aJKWKiJ3ZAschIwEYhjkccjN3UvFej6Tf/Y7y4l84KHkENtLMsCno0rIpWJTg8uVHB9DQBsUVxXiXx6bDwVD4r8Pw/bdLhuo/tvn28sTtbFtrPEGCkkEqQcFSAcV2cciSxrJGwZHAZWB4IPQ1bg1FT6P+v1C+th1FFFQAUUUUAFFFFABRRWF408TR+EPB+oa3JH5zW0Y8qH/nrIxCov4sRTjFyait2Bu0V55rfxH1LwbeeHrbxZosYg1GFftmpWtxiO3mx+8AiILbF4O4kcHvjFa9l41ml8Za7pOoaZHYabo0CTSapJeoVYOAy5TA2jG45ycbeetdzy7EqHtOX3Wr3utUnbTXe7StvrsRzx/L8TrKK5nw/wDEXwn4pup7bQtahup4FLPFsdGKjqVDAbgPUZqjYfF7wJqd9aWdl4hhee8fZCjQypls4AJZQFJPTdjPGOtT9Qxl2vZSut/denrpoPnja9ztKK5vxH8QvCvhO+is/EGsw2lzMAVi2O7AHoSFB2j3OBW/a3UF9aQ3VnMk9vMgkiljbcrqRkEHuCKwnQqwgqk4tRezadn6PqO6vYlooorEYUVlQX1+fETWV2lukLW7SxiMszDDhcljgcg9MceppLy71K41KWz0h7WI20aSSvcxNIHLFsIArKVPy5LHP3hweajnVrnR7CV7Nra9+n9XNaisZ9Zlu9N019NVI59T/wBWZl3LCNhcllBBbGMYBGSRzirOlXk8/wBptr4xNdWcojkeFCqPlQwYAkkcN0yeQeaakm7Clh5xi2+nTrva/wB+hoUUUVRgFFZWq31/ZXVp5CW/2aW4jidnLM7bjggAYC465yfp3qXVLy4ga1trHyxc3chjR5VLJGAjMWKggsPlxgEckc1PMrM3VGT5fO/4bmhRWKmtTWul6k+oLHNdabkSCAFFmOwOu0MTtzuAwScHPOKktLvULfVY7LVpLWU3MbywtbQtHs2Fco2Wbd94YbjoeBxS50U8PNJv+npfT5a6mtRXL+GfFF3qXijxD4f1iCGG90mdWhaDIWe2kGYn5/iGCGxxkV1FbTg4Oz/q5zdbBRRRUAFFFFABRRRQAUUUUAFFFFABRRVXU71dM0m7vnQyLawvMUXqwVScfpSbSV2VGLlJRjuy1RWPY3mowXdvba3LZySXaM0X2WJkCFQCVO5m3cE/N8vTpzVa51LW7FlvLyOzWze6WBbNVYz7WkCK/mbtpPIYpt6ZG7ip50joWFk5cqa8vP0+5+XnsdDRXM65r81j4gSwGs6RpUZtllVtRjLGVizLhf3qdMD161sahqI0nRZL26/fvGg+WFcedIcBVUEnlmIAGT1HNJVE032FLDVEod5bb/5W+5l6isDR9R1nV/CIuVFlDqplliO5XMKlJmQnAOTwvr19KfYXWrr4jbT9Qu7C6jW089zb2jwtGS+1Osr5B2yen3RT59UrblSwsoucXJXje69Plb011Nyiuf8A7Yv9o1Mm2Gl/afs/k+Wxmx5vlCTfuxjODt25x3zxVq8u9SuNSls9Ie1iNtGkkr3MTSByxbCAKylT8uSxz94cHmlzqwnhpp6teflbp+K27mtRVbTb1NS0u2vY1KrPGsgU9VyOn4VZrS99Uc8ouMnF7oKKKKCQooooAKKKKAKiLvbbkjIPI6jjr+v+c8+daf4Nns7GHRbzQdevWgURC6XxROtjKq8KzJ9o3pkDJQRMAeBkc16PD/rR/n/PX9ffmzQtHcHqrHnzWmtvqd5P/wAI5fKi+Ior1MzW2ZoRAIiy/veMFc4bBwwwCcgVfsfiW1utGFv4VnuP7L1a/u5JJbm2VZYpDPtMR8wneRKMBgo4IJXrXpdFC0Vv62S/Qbd3f+uv+ZwWpeHdV8Uw6tfzWMmkzXEdpHbWct2I5ZPs8zS5klgZtm4ttBRiVHPU4E2gaHnWYLl9A1+xeBXBn1bxDJdKhZSuYo/PlDE5xlgmBz14rt6KBPU821Ox8S/8IBH4TtvDEk8lnBDF9tW6gWCdYipBjBffvbaPlZVAJPzEAE7iSa1oupalPaeHbjUk1aZLpDHcQRm2fyY4zHNuccDy87o9/BIxwN3W0U763Y7nBaRHrmjfDSLSJfDF/cXwSe2aK1mtdoyWKyAvMo2Hdx/Fxyop14NZ1D4U32kHwtqUV8NPWyS2lmtD5zFNhZSJiu0dTuIOOgPSu7opd/O34f8ADi6p9jlbsajc+LtBni0G+itEsp0nm823AtWk2YRgJckr5ZzsDDkYJ5xzuqjVtO8FaX4dutECJY3On251BpojBMEniCmJAxk3nA4ZVA+blsDPplZg8NaEutnWV0XThqjHJvhaJ55OMf6zG7px16U09f6/r+uodLI06zPEF1dWukyCx0m71SWUGPyrR4VZcqfmJlkQY+hJ56VSuPDmqTXMssfjTXLdHcssMUNiVjBP3RuticDpySfc1o6Tp11p0Dpeaze6szNkSXiQKyDHQeVGgx9QT71LXNGzGnyu6M/wW18nhWws9U0i70ueytordkuZIX8wqgBZTFI4xkd8H2rfooqpScndkxioqyPPvGgbRvid4Q8SSqzWLNLpVy4UnyWmx5TewLjaT7iotU0dpPiPfaxf+ArzWVjjsxp99C9mHjkhaRyymSdHUZZeOM7TkY6+jUVcp80Yr+X/ADv+rH1b7/1+VvuPN/s/ivUpUOq6DeefNpOpWzTtLahYmll3QowWXOQkaqSoIyw5PzEQazpniK60PWLW28MXcsuqaFbWcYkntdkUyGUMsn77p+8U5XcMZ78V6fRWa02/rVv9R3/r7v8AI5Ly9Z8Q6tpQv9Dl0eLS52uJLiSeKQSt5TxhYdjFtp3klnVDgAbeTjnNP8Gz2djDot5oOvXrQKIhdL4onWxlVeFZk+0b0yBkoImAPAyOa9Qoo63E9VY5VzrGgazqz2OhTaxFqlwtzFLBcQx+SwijjKy+Yynb+7BDIHOCRt4GcXw7pWvaFpei3kuhSz3GmQXVlPYxTwh3WSVHEsJLhNvyAYZkOD0yMH0Sij+v6+Q7nFaG+u2mveItVv8AwxdxrfJBNbww3Fu7tsXZ5bEyACTHPXZjgMTU3hS71eG+1GPUPC2qWMd3eS3STzTWjIqlVwpCTs24lccAj3rr6KNnddrCev33POltNaX4dy2ieFb9b59Ya4NqJrQOUN39o8zd52z7vy/ezu7Y5rZZtW0PUtWlt9Dk1CHVZluklFxDGts3kpGVn3uDtHlg7kD8EjHAz1lZuq+G9D12SGTXNG0/Ungz5TXlqkxjzjO0sDjoOnpQO55vrlzJD+zrpmk20Imv9as4NNs4Y0OHeUDkdcKFDNn0Feo6ZZjTtJtLJXLi2gSEMerbVAz+lWAAqhVAAAwAO1LWsql4cqXVv+v66k22/r+ugUUUVkMKKKKACiiigArivi3pd3qnw6vG02Iz3VjLFfRwgZ83ynDlcd/lB49cV2tFXTm6c1NdHf7g02Z5xr8V54017wfqOmaEuq+HJYJJLq4kuUjAjuI9hBjb5jhTnjr04rBsvhh4kGj+MNGu7iOZJxaJpV1csrLcxwMWSOQA5AACoSQPUZr2WivVp5tWo01TpJJK1t3tLmT33vvok102M/Zpu7/r/hzzay0/xZ4j8WaJqOu+GLLw/BoPmOjR3iTyXTNGUCIVH7tOckH0Fc8fD3jJ/hrqen/8IYialea59t8oajBkoZfO3Fs4OCoTrnnPbFe1UU4ZvODXLTjZWsve0s2/5r6ttu7flYXs01q/6tY8O8UeA/GD+LL3XdLtNZmXWYopJ7bSvEMdk1s4jCtHIXQiQA5wR2zXoHhmD/hAfhQp1O3ltl0u1mnkglu1uHVQWfaZFRAT9F9uep7Go7m2gvLd7e7hjnhkGHjlQMrD0IPBqMTmlXFUI0KkUlG2qvfRWXVrReRUYJS5ipoV/c6poFjf39l9guLqBJZLXzN5hLDO0tgZIzzwOav0UV5Umm20rFK6WpzjXOonxYtwPD+ofZ1ha287zbfHLg78ebnbgZ6Z9qtXS3mnaxPeWWnzX63kUaFIpI08p03ctvYfKQw6ZI29DmtmislCytc7Hib29xWta2uvXv37WMBdJuNN0nRzCjXk+lgB4oiqmYGMq23cQByQRkjgYzVzR7e4WW+vruFreS9mEggYqWiVUVAGKkgn5SeCRyB2zWnRTUEndEzxM5xalu+vzvbtvrtf5GF4a8Rv4jk1Z1svs9pY38llBOZd32ny+HcDAwN2VHJztPSt2o7e2gtIRDaQxwRAkhI0CqCSSTgepJP1NSVrJpv3VY5jA8Ry37S2kVlo15erFcRTtLDJAq4VslfnkU5/DHPWrN+lzMmn6lDZTGe1cyNZloxIyshUrnds3DIP3scYzzWtRWXJvrudKxDUYxUVpfvrfe+v5WMBdJudR0zWDcobObU2zHE5VjDiNVXdtJBOVycEjnHbNS2q3uo63Be3eny6fHaQyRhJnjYys5TkbGb5QEPXBORwMc7VFHIhvFSaasvLy0tpr2Vtb/eefeEg2tfFrxX4hhRlsbaOLR4JCCBNJGS0p99rHbn64r0Giit6k+drySX9fmcltWwooorMYUUUUAFFFFABRRRQAUUUUAFRXP8Ax6y/uDcfIf3Ix+84+7zgc9OeKlopNXVhp2dzltF8OwRatBqKaXPpy28bJGl3c+fLk8YHzuEjAzhVbknoMcthudZm1JbrUPDd88yy7YE+0Wwgt0JwXz5hZm25Oduf4QByT1dFZ+zWlmdzx05ScpxT0trzaLsne+t9dfLYzL+/1C1mZINFlv4mX5HgnjHPcOHK4HuN30rObw3K/hvS4J7q5hutNAmUWJjOZApG0CVSpA3EDIHY8dukopuCe5lDEyppKCS/XRrW7a6s57wTZX+n+H3g1WO6jm+1zyBbpoS215C4I8rjndk553FscYq/plpNHfaleXUYSS5nAjGQf3SKFXp6nc3/AAKtKiqUUreQquJlVnObSXPvb1vp8zmTp999lGhrZS/Zxdib7eXj8vyxN5u3bu37sfL93HfPart0t5p2sT3llp81+t5FGhSKSNPKdN3Lb2HykMOmSNvQ5rZoqfZop4qTesVZ3utdb2u979FtZFLR7FtM0WzspHEjwQqjOowGYDkgemau0UVaSSsjmnJzk5S3YUUUUyQooooAKKKKAK0P+tH+f89f19+bNUZLhLWJ7iUSMkSF2EUTSuQBnhVBZj7AEnPv82T/AMJ5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVzf/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFAHSUVDaXUd7ZxXMKyrHKoZRNC8Tge6OAyn2IBqagAooooArQ/wCtH+f89f19+eJ1DxzfXPjS40TRmksks1iEktz4bvrvzXdnXAKFFRBs4ckq2Tg4U120P+tH+f8APX9ffnk7K61q38f6teSeEtW+x3sdtbR3Ans9o8tpd0hHn7tuJARwWxngHihbj6M1devtZ0zUtLltbmxNhcXUVrPBJaOZSXJBZZBKAo6cFG78+mnqmrW2j2qz3iXTozbQtpZy3L5xn7kSs3briuf8X3OrSXVjb6d4Y1PUI7W9humuIJrVUZVOSoEkytn6qB71H4kGs6pHpc66Rrf2BlkN5p1hqENrdrJx5ZaRZlBQAPkJKOSuc8gJfD8/0X/BD7Xy/wA/+AbQ8TabJoM+rWzXFxBASskcVpK8yOMZRoQpkVhkZBXIHJGKq+F/GNn4osrWW3tNRtpZrVLhluNOuYo1yASFleNUfrxg8jkcVneB9M1PT7fXbXUdLubFbi78+3kuNQ+2b0aFEwZGYyFlKHO4YGQFLDmpvCdxr1pp+laLe+HJbSKxtUt7m9mvITG5RNoMSoWZgSB98R4Bz1GKpLf0X5ah0F0r4g6dqmp3VkLDWInhuxaozaPebWyiNudjCBHy+MMRgAN0IrS1LxXpWlX/ANiuWu5Jwod1tLCe5EQPQuYkYJnB+8RwM1lLLr+i61rMdj4ck1FdQvFube6F5FFAgMMaESFj5ikGMn5Y34I9wM7VvDVxB4o1G+fTNe1S31GRJVbSNektDCwjVCrxefEhGEBDAljkggYBK6IbtdnSal4x0PSY7KS9vH8q/Utayw28kyTcbgqsikFiPurnLfwg1c0fW7DXbV7jTJmdYpDFKksTxSROACVeNwGQ4IOCAcEHvXMPpd5YyeEYLTw7cyW9leSTz7L1Lj7KHjlQMzzOHdsyhjgNjDYJ4zp+H0v18UeI5LzSbmzt7i5je3uJZIWWZViWMkBHZhyhPzAcEd8gMnodHWDBquqjxoNLvorOO1ls5biIQl3cbJEUEscDkPnaF4I+8a3q42a91k/ECC9XwlqzWcVrLZm4E9ntJaWMiQAz7tmEJ6buny0l8S+f5P8AUUvhdvL81+h2VIzBFLMQFAySewpahvLYXlhcWrMVE0TRlh1GRjP60FK19TkvDHjDVPEPgvU/EqWFr9n82dtLhaRozNBHwGkbDYLFWPCnjHXrSaZ4u8R30WnXMnh/Szb3f2Z5UtNXknuLeObG12j+zKMAZJywHynnisn4eT3Q+EFzoRspJdW0RZ9MntI9qM0i527S5VcMrK2Scc9a2fhxoNtoehKieFn8PXht4IrsHyAty6JgyAQyMOpbk4Y55retGMa00tla3mtfzVjPVpd9b+X/AA2po2d/rEfjWfTNRuLGezktWurfyLV4pIwJAoV2MjB+D1Cr9Ktar4m07RrhYb9b4ErvMkOnXE0aL6tIiFUHBzuIwOawG1HW/wDhNhqA8GaybZbM2u8T2PJ8wNux9oztwPr7VW8QaXrd7r9+JdN1m/ik2jTpbLWzY2sC7BuEwSRXLb9xJ2ScFQMciua75E/X83b8LGlryfy/Jfrc3fEfjGx8PafY3hhur6K+mijiaztZp1KO6rv3RIw6PkA4L9Fyasr4nsZNDuNVjt9T8i3O1o5NKukmJ4+7E0YkYcjkKR19DjCi03V2+F+iWv8AZUsepab9iaSxlnj3v9nkQsFcMUywQlcsOozt5xvaZfapqsV19v0WTSYtoWBbm5jklckHJZYyyqOmMOxOTkDHNTVlK3S9vwBW0v8AMq+F/GNn4ps7WS3tNRtpprVLhluNOuYo1yASFlkjVH+9xg8jkcVJF4z0SbUPsiT3AJl8lbhrGdbZ3zt2rOU8pju4ADHJ461n+FZtettM0vQ73w7NZxWVoltc30t5CUYpHtDQqhZmyQPviPAOeoxXOaf4Nns7GHRbzQdevWgURC6XxROtjKq8KzJ9o3pkDJQRMAeBkc1TtzeX9f1qJ6I6+/8AHOg6ZqVxp93Ndi7twrSQxadcSNsIJ3gLGdyDBy4yoPBIPFVPGni6fw/4Qt/E2jpb6hpqSwy3bAli1o5wZIipwSNytzxjNSW4v7fx5rl82gXjW76fBHBcrJb4uGiMjFFBk3AnzQBuCjKnJHBPI+JpNQsfgNBoNzpc9vrOpRx6Rb2cpjdmlc4JzGzLgKGbOe3OK0oR5qkU1fVf1/Ww3b5f1+R6pHIksayRsGRwGVgeCD0NOqtplmNO0m0slcuLaBIQx6ttUDP6VZrKVk3YmN2lcKKKKQwooooAKKKKACiiigAooooAKwvGniaPwh4P1DW5I/Oa2jHlQ/8APWRiFRfxYit2uK+Lel3eqfDq8bTYjPdWMsV9HCBnzfKcOVx3+UHj1xWlFRlUip7Nq/pcav03KOt/EfUvBt54etvFmixiDUYV+2ala3GI7ebH7wCIgtsXg7iRwe+MVr2XjWaXxlruk6hpkdhpujQJNJqkl6hVg4DLlMDaMbjnJxt561ia/FeeNNe8H6jpmhLqvhyWCSS6uJLlIwI7iPYQY2+Y4U5469OKwbL4YeJBo/jDRru4jmScWiaVdXLKy3McDFkjkAOQAAqEkD1Ga+jhh8BKgnV5Yztrq+s7XtfRxjq0+jvbRnPeV1bbT8v169jv/D/xF8J+Kbqe20LWobqeBSzxbHRio6lQwG4D1Gao2Hxe8CanfWlnZeIYXnvH2Qo0MqZbOACWUBST03YzxjrWVZaf4s8R+LNE1HXfDFl4fg0HzHRo7xJ5LpmjKBEKj92nOSD6CuePh7xk/wANdT0//hDETUrzXPtvlDUYMlDL524tnBwVCdc857YpRy/AOVpSt8P/AC8hpdtN3taVkk9LWvZ66g5ztp+T7f16302PRvEfxC8K+E76Kz8QazDaXMwBWLY7sAehIUHaPc4Fb9rdQX1pDdWcyT28yCSKWNtyupGQQe4IrxPxR4D8YP4svdd0u01mZdZiiknttK8Qx2TWziMK0chdCJADnBHbNegeGYP+EB+FCnU7eW2XS7WaeSCW7W4dVBZ9pkVEBP0X256nkxmCwlHC06lGpzTdrq8X010Wqs9Nb33LjKUp2todjRVDQr+51TQLG/v7L7BcXUCSyWvmbzCWGdpbAyRnngc1frxZRcW4voWmmroyoL6/PiJrK7S3SFrdpYxGWZhhwuSxwOQemOPU0l5d6lcalLZ6Q9rEbaNJJXuYmkDli2EAVlKn5cljn7w4PNUmudRPixbgeH9Q+zrC1t53m2+OXB3483O3Az0z7VaulvNO1ie8stPmv1vIo0KRSRp5Tpu5bew+Uhh0yRt6HNc6bcevX9f+Aem4RUk1y35dNVa9+uttu/5g+sy3em6a+mqkc+p/6szLuWEbC5LKCC2MYwCMkjnFImtTWul6k+oLHNdabkSCAFFmOwOu0MTtzuAwScHPOKiXSbjTdJ0cwo15PpYAeKIqpmBjKtt3EAckEZI4GM0LpNzqOmawblDZzam2Y4nKsYcRqq7tpIJyuTgkc47Zo9/5/wDA/wAyuXD+XLf5/F99uX5fMs2l3qFvqsdlq0lrKbmN5YWtoWj2bCuUbLNu+8MNx0PA4qr/AGxf7RqZNsNL+0/Z/J8tjNjzfKEm/djGcHbtzjvniprVb3UdbgvbvT5dPjtIZIwkzxsZWcpyNjN8oCHrgnI4GOaZ0+++yjQ1spfs4uxN9vLx+X5Ym83bt3b92Pl+7jvntR73T5f8H8fkOMaXMua19L7Wtd3t0va22v4jdT8UXej/ABF0jRryCH+y9YgkW2uFz5iXMfzFG7bWTp3yDXUV594wDa78U/COiWiMx02R9YvJQDiKMKUjGf8AackYz0Feg13VIpQg+rX6vU8n7TXp/X5P5hRRRWAwooooAKKKKACiiigAooooAKKKKACiiigAqrqd6NN0u5vGQyeRGzhAQC5A4XJ9TxVqqer2J1LR7qzV/LaaIqrkcK3Y/nilK9nY0pcvtI8+11f0Ktpd6hb6rHZatJaym5jeWFraFo9mwrlGyzbvvDDcdDwOK1qxbVb3UdbgvbvT5dPjtIZIwkzxsZWcpyNjN8oCHrgnI4GOdqphexriEk1te2trWvd9tNrbBRRRVnMFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAVof8AWj/P+ev6+/Nmq0P+tH+f89f19+bNABRRRQAUUUUAFFFFABVHVbC51G2WK01e80p1fcZrNIWZhj7p82NxjvwM8davUUAc9D4a1WKdHfxtrsyqwJjeCwCuAehxbA4PsQfeuhoooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKjubaC8t3t7uGOeGQYeOVAysPQg8GpKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAKMlwlrE9xKJGSJC7CKJpXIAzwqgsx9gCTn3+bJ/4TzSP+fPxB/wCE5qH/AMYrbh/1o/z/AJ6/r788n4l+IFpZ2Mv9iXifaYZgvmXOnzvbT4bDRpMNsZf6O2CCNp7G7sHS5o/8J5pH/Pn4g/8ACc1D/wCMUf8ACeaR/wA+fiD/AMJzUP8A4xVvUvFelaVf/YrlruScKHdbSwnuRED0LmJGCZwfvEcDNN1Lxjoekx2Ul7eP5V+pa1lht5Jkm43BVZFILEfdXOW/hBoDyK3/AAnmkf8APn4g/wDCc1D/AOMUf8J5pH/Pn4g/8JzUP/jFaFh4k0nUdKudRt7vZbWhYXJuI2ga3KjcwkSQBkIUg4YDgg9DUGm+LtI1W9itLV7tJ5lZokubCe381VAJZTIihhyORxzQBW/4TzSP+fPxB/4Tmof/ABij/hPNI/58/EH/AITmof8Axip4vGeiTah9kSe4BMvkrcNYzrbO+du1ZynlMd3AAY5PHWt2gPI5v/hPNI/58/EH/hOah/8AGKP+E80j/nz8Qf8AhOah/wDGK1dYm1CDT3k0pLUyqCWe5ZtqAAnIVR85zjjK/WjQb+XVPDem6hcKiy3dpFO6oCFDMgYgZzxk0LW/l+v/AAwPS3n+n/DmV/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFdJXLeKvFF3pWveHtD0aCCbUNYuiG8/dtit4xumfjnOOF7ZNXCDnLlQPRXJf+E80j/nz8Qf+E5qH/wAYo/4TzSP+fPxB/wCE5qH/AMYpl9rviNfFVxpGkaJpdzHDbR3Kz3WqyQFlcsuNq274IKN3PGD3wLC+MdNj0iG9vVuYJZJnt2s4rd7idZkJDoEiVmbG0nIBGMHoajoH9fqRf8J5pH/Pn4g/8JzUP/jFH/CeaR/z5+IP/Cc1D/4xVqTxdo0Wgz6w884s7Zttx/oc3mwHj78WzemAQTuUYByeOaXSfFmja3fNaaddO84jMqrLbyRCaMHBkjLqBImSPmTI5HPIo8g8yp/wnmkf8+fiD/wnNQ/+MUf8J5pH/Pn4g/8ACc1D/wCMVLP430G3u5oJLm4/cy+TJOljO8Cybguzzghj3biAV3ZB4qbUvFelaVf/AGK5a7knCh3W0sJ7kRA9C5iRgmcH7xHAzQBU/wCE80j/AJ8/EH/hOah/8Yo/4TzSP+fPxB/4Tmof/GKg8a+LLjw54Tg8SaTHb3+nRTRSXhBLFrVzgvEQcFhuU88YzXUxyJLGskbBkcBlYHgg9DVuElHm6bf194rnO/8ACeaR/wA+fiD/AMJzUP8A4xR/wnmkf8+fiD/wnNQ/+MV0lFQM5v8A4TzSP+fPxB/4Tmof/GKP+E80j/nz8Qf+E5qH/wAYrpKKAOb/AOE80j/nz8Qf+E5qH/xij/hPNI/58/EH/hOah/8AGK6SigDm/wDhPNI/58/EH/hOah/8Yo/4TzSP+fPxB/4Tmof/ABiukooA5v8A4TzSP+fPxB/4Tmof/GKP+E80j/nz8Qf+E5qH/wAYrpKKAOb/AOE80j/nz8Qf+E5qH/xij/hPNI/58/EH/hOah/8AGK6SigDm/wDhPNI/58/EH/hOah/8Yo/4TzSP+fPxB/4Tmof/ABiukooA5v8A4TzSP+fPxB/4Tmof/GKP+E80j/nz8Qf+E5qH/wAYrpKKAOb/AOE80j/nz8Qf+E5qH/xij/hPNI/58/EH/hOah/8AGK6SsLxp4mj8IeD9Q1uSPzmtox5UP/PWRiFRfxYinGLk1FbsCD/hPNI/58/EH/hOah/8Yo/4TzSP+fPxB/4Tmof/ABisDW/iPqXg288PW3izRYxBqMK/bNStbjEdvNj94BEQW2LwdxI4PfGK17LxrNL4y13SdQ0yOw03RoEmk1SS9QqwcBlymBtGNxzk4289a7nl2JUPacvutXvdapO2mu92lbfXYjnj+X4lj/hPNI/58/EH/hOah/8AGKP+E80j/nz8Qf8AhOah/wDGKb4f+IvhPxTdT22ha1DdTwKWeLY6MVHUqGA3AeozVGw+L3gTU760s7LxDC894+yFGhlTLZwASygKSem7GeMdan6hjLteyldb+69PXTQfPG17mh/wnmkf8+fiD/wnNQ/+MUf8J5pH/Pn4g/8ACc1D/wCMUeI/iF4V8J30Vn4g1mG0uZgCsWx3YA9CQoO0e5wK37W6gvrSG6s5knt5kEkUsbbldSMgg9wRWE6FWEFUnFqL2bTs/R9R3V7GB/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFdJRWIzm/wDhPNI/58/EH/hOah/8Yo/4TzSP+fPxB/4Tmof/ABir8F9fnxE1ldpbpC1u0sYjLMww4XJY4HIPTHHqazNT8UXej/EXSNGvIIf7L1iCRba4XPmJcx/MUbttZOnfINOmnV+Hz/D/AIYupB09/Uk/4TzSP+fPxB/4Tmof/GKP+E80j/nz8Qf+E5qH/wAYrpKKRBzf/CeaR/z5+IP/AAnNQ/8AjFH/AAnmkf8APn4g/wDCc1D/AOMV0lFAHN/8J5pH/Pn4g/8ACc1D/wCMUf8ACeaR/wA+fiD/AMJzUP8A4xXSUUAc3/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFdJRQBzf/CeaR/z5+IP/AAnNQ/8AjFH/AAnmkf8APn4g/wDCc1D/AOMV0lFAHN/8J5pH/Pn4g/8ACc1D/wCMUf8ACeaR/wA+fiD/AMJzUP8A4xXSUUAc3/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFdJRQBzf/CeaR/z5+IP/AAnNQ/8AjFH/AAnmkf8APn4g/wDCc1D/AOMV0lFAHN/8J5pH/Pn4g/8ACc1D/wCMUf8ACeaR/wA+fiD/AMJzUP8A4xXSUUAc3/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFdJRQBzf/CeaR/z5+IP/AAnNQ/8AjFH/AAnmkf8APn4g/wDCc1D/AOMV0lMmMqwubdEeUD5Fdyqk+5AOPyNA1q7HPf8ACeaR/wA+fiD/AMJzUP8A4xR/wnmkf8+fiD/wnNQ/+MVpaNe3V5Hdi+8nzbe5aH9ypC4AB7k+vXv6CqP9sX+0amTbDS/tP2fyfLYzY83yhJv3YxnB27c4754qOdWT7nQsNNyce34t7JepH/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFXry71K41KWz0h7WI20aSSvcxNIHLFsIArKVPy5LHP3hweaY+sy3em6a+mqkc+p/6szLuWEbC5LKCC2MYwCMkjnFHOhLDzaT/paX1+WpU/4TzSP+fPxB/wCE5qH/AMYo/wCE80j/AJ8/EH/hOah/8YrRsL28mhvLeZYJb+zfy2Kboo5SVDKf4ioIYAj5sEHrS6Ne3V5Hdi+8nzbe5aH9ypC4AB7k+vXv6CmpJuyFKjKKb7W/HZmb/wAJ5pH/AD5+IP8AwnNQ/wDjFH/CeaR/z5+IP/Cc1D/4xXSUVRgc3/wnmkf8+fiD/wAJzUP/AIxR/wAJ5pH/AD5+IP8AwnNQ/wDjFdJRQBzf/CeaR/z5+IP/AAnNQ/8AjFH/AAnmkf8APn4g/wDCc1D/AOMV0lFAENpdR3tnFcwrKscqhlE0LxOB7o4DKfYgGpqKKACiiigCoi7225IyDyOo46/r/nPPA6nY+Jf+EAj8J23hiSeSzghi+2rdQLBOsRUgxgvv3ttHysqgEn5iACfQIf8AWj/P+ev6+/Nmgd9Dz3UPDtyniG91GbR/EN9DqbRzhNL157R7ZxGqMkkYuI4zjYDuUsTkjoATcfS7yxk8IwWnh25kt7K8knn2XqXH2UPHKgZnmcO7ZlDHAbGGwTxntqKFoLrc4+GLUzfeLfP8MzTw3kqfZ47qeARXqCJImHDsVB2k/Mo4I75AZoGnalBroFrp+saXpgikW5XVNT+1+c52+WYR50pQD5v7nGBtP8PZ0UDueX6f4Nns7GHRbzQdevWgURC6XxROtjKq8KzJ9o3pkDJQRMAeBkc16hRXP3HhzVJrmWWPxprlujuWWGKGxKxgn7o3WxOB05JPuaLsXW5d8QXV1a6TILHSbvVJZQY/KtHhVlyp+YmWRBj6EnnpVPwW18nhWws9U0i70ueytordkuZIX8wqgBZTFI4xkd8H2rQ0nTrrToHS81m91ZmbIkvEgVkGOg8qNBj6gn3q/QtL+dvw/wCHB628r/1+AV5940DaN8TvCHiSVWaxZpdKuXCk+S02PKb2BcbSfcV6DRWlKfs5c3r+KsJ6pr+v6ucheXWr2Hj27vLfwvqeoWj2MNsk9vNaKrMru5OJJlbA3gdOoPGME5F74Mv0Nrq0tteXc5nu5rux0zVZLORRO6MPLkV4w5Xy1UhmVW5bggCvRqKzKvd/12see3ehXQ8DeIotN8N6qLrU7b7OlvfauLu5kJVl+YyzMiKu7tIcjPGcA7NwdRm8ZeH7mLQL2O0js5lnlMluFtmk8vCMBLkkeWQdgYcjBPOOpop3F0scBLpurHUhNpPh7U9I1Ca5Ej3C6sraeoMmZGaAS4ZmXcTiLJZvvA/NRq3hqeDxRqN8+ma9qltqMiSq2ka9JaGFhGqFXi8+JCMICGBLHJBAwCe/oqR3PNfHsbaT8HW0CzsGjvtXYadZ2ZuXumMkzkndK2SSF3MSeAQRk8GvQdMsxp2k2lkrlxbQJCGPVtqgZ/SrNFbOpeHJ53/r+upNtvL+v8gooorIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFcV8W9Lu9U+HV42mxGe6sZYr6OEDPm+U4crjv8oPHriu1oq6c3Tmpro7/cGmzPONfivPGmveD9R0zQl1Xw5LBJJdXElykYEdxHsIMbfMcKc8denFYNl8MPEg0fxho13cRzJOLRNKurllZbmOBiyRyAHIAAVCSB6jNey0V6tPNq1GmqdJJJWtu9pcye+999Emumxn7NN3f9f8OebWWn+LPEfizRNR13wxZeH4NB8x0aO8SeS6ZoygRCo/dpzkg+grnj4e8ZP8NdT0//AIQxE1K81z7b5Q1GDJQy+duLZwcFQnXPOe2K9qopwzecGuWnGytZe9pZt/zX1bbd2/KwvZprV/1ax4d4o8B+MH8WXuu6XaazMusxRST22leIY7JrZxGFaOQuhEgBzgjtmvQPDMH/AAgPwoU6nby2y6XazTyQS3a3DqoLPtMiogJ+i+3PU9jUdzbQXlu9vdwxzwyDDxyoGVh6EHg1GJzSriqEaFSKSjbVXvorLq1ovIqMEpcxU0K/udU0Cxv7+y+wXF1Akslr5m8wlhnaWwMkZ54HNX6KK8qTTbaVildLU5xrnUT4sW4Hh/UPs6wtbed5tvjlwd+PNztwM9M+1YPjANrvxT8I6JaIzHTZH1i8lAOIowpSMZ/2nJGM9BXoNFOhL2Lvvv8Aj/VzetU9rFLlSsraX/VsKKKKkxCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACo55Ght5JI4XnZVJEUZUM59BuIGfqQKkooGtGYHh6W/F3fJeaNeWSXFw86SzSQMuCFAU7JGOeD2x71AdPvvso0NbKX7OLsTfby8fl+WJvN27d2/dj5fu4757V01FZ+zVkr7HX9blzuailt33Wz3/4HkY10t5p2sT3llp81+t5FGhSKSNPKdN3Lb2HykMOmSNvQ5qFdJuNN0nRzCjXk+lgB4oiqmYGMq23cQByQRkjgYzW/RT5ESsVJJKy8/PS2uvZ20t95kadHdQJqOpT2comu5BKlmrIZFVY1UKTu2bjtJ+9jkDPGar+Hpb8Xd8l5o15ZJcXDzpLNJAy4IUBTskY54PbHvW/RQoWd0weIbUk4rW3fS21tfzuFFFFWcoUUUUAFFFFABRRRQAUUUUAZ88xgtpplZFaONnDOCVGFJycZOPp/Xnwb/hfev/8AQyeA/wDwB1X/AON19AQ/60f5/wA9f19+bNdNCrTp354c39ejE9VoeE+IvjVrWk67PZw6/wCC4kjWMhLmz1JpBuRW5KIV78Y7YzzRe/GrWrfw/pd8mv8AgtZLtpg7yWepGN9jADYAm4Yzzu9sV7tRW6xNCy/d7ea/+RJs+54TYfGrWrrQtVvH1/wWz2axFHjs9SEabn2neCm457be/Xijw78ata1bxBaWM+v+C5o5mIKWtnqSyH5SflLoF7d692ooeJoWf7vfzX/yIWfc+dv+F96//wBDJ4D/APAHVf8A43Wn4i+NWtaTrs9nDr/guJI1jIS5s9SaQbkVuSiFe/GO2M817tRT+s4e9/Zfiv8A5EOV9zwm9+NWtW/h/S75Nf8ABayXbTB3ks9SMb7GAGwBNwxnnd7YosPjVrV1oWq3j6/4LZ7NYijx2epCNNz7TvBTcc9tvfrxXu1FL6zQtb2f4rv/AIQs+54T4d+NWtat4gtLGfX/AAXNHMxBS1s9SWQ/KT8pdAvbvWZ/wvvX/wDoZPAf/gDqv/xuvomin9Zw97+y/Ff/ACIcr7nhPiL41a1pOuz2cOv+C4kjWMhLmz1JpBuRW5KIV78Y7YzzRe/GrWrfw/pd8mv+C1ku2mDvJZ6kY32MANgCbhjPO72xXu1FJYmhZfu9vNf/ACIWfc8JsPjVrV1oWq3j6/4LZ7NYijx2epCNNz7TvBTcc9tvfrxR4d+NWtat4gtLGfX/AAXNHMxBS1s9SWQ/KT8pdAvbvXu1FDxNCz/d7+a/+RCz7nzt/wAL71//AKGTwH/4A6r/APG60/EXxq1rSddns4df8FxJGsZCXNnqTSDcityUQr34x2xnmvdqKf1nD3v7L8V/8iHK+54Te/GrWrfw/pd8mv8AgtZLtpg7yWepGN9jADYAm4Yzzu9sUWHxq1q60LVbx9f8Fs9msRR47PUhGm59p3gpuOe23v14r3ail9ZoWt7P8V3/AMIWfc8J8O/GrWtW8QWljPr/AILmjmYgpa2epLIflJ+UugXt3rM/4X3r/wD0MngP/wAAdV/+N19E0U/rOHvf2X4r/wCRDlfc8J8RfGrWtJ12ezh1/wAFxJGsZCXNnqTSDcityUQr34x2xnmi9+NWtW/h/S75Nf8ABayXbTB3ks9SMb7GAGwBNwxnnd7Yr3aiksTQsv3e3mv/AJELPueE2Hxq1q60LVbx9f8ABbPZrEUeOz1IRpufad4Kbjntt79eKPDvxq1rVvEFpYz6/wCC5o5mIKWtnqSyH5SflLoF7d692ooeJoWf7vfzX/yIWfc+dv8Ahfev/wDQyeA//AHVf/jdafiL41a1pOuz2cOv+C4kjWMhLmz1JpBuRW5KIV78Y7YzzXu1FP6zh739l+K/+RDlfc8JvfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YosPjVrV1oWq3j6/4LZ7NYijx2epCNNz7TvBTcc9tvfrxXu1FL6zQtb2f4rv8A4Qs+54T4d+NWtat4gtLGfX/Bc0czEFLWz1JZD8pPyl0C9u9Zn/C+9f8A+hk8B/8AgDqv/wAbr6Jop/WcPe/svxX/AMiHK+54T4i+NWtaTrs9nDr/AILiSNYyEubPUmkG5FbkohXvxjtjPNF78atat/D+l3ya/wCC1ku2mDvJZ6kY32MANgCbhjPO72xXu1FJYmhZfu9vNf8AyIWfc8JsPjVrV1oWq3j6/wCC2ezWIo8dnqQjTc+07wU3HPbb368UeHfjVrWreILSxn1/wXNHMxBS1s9SWQ/KT8pdAvbvXu1FDxNCz/d7+a/+RCz7nzt/wvvX/wDoZPAf/gDqv/xutPxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5r3ain9Zw97+y/Ff/IhyvueE3vxq1q38P6XfJr/gtZLtpg7yWepGN9jADYAm4Yzzu9sUWHxq1q60LVbx9f8ABbPZrEUeOz1IRpufad4Kbjntt79eK92opfWaFrez/Fd/8IWfc8J8O/GrWtW8QWljPr/guaOZiClrZ6ksh+Un5S6Be3esz/hfev8A/QyeA/8AwB1X/wCN19E0U/rOHvf2X4r/AORDlfc8J8RfGrWtJ12ezh1/wXEkaxkJc2epNINyK3JRCvfjHbGeaL341a1b+H9Lvk1/wWsl20wd5LPUjG+xgBsATcMZ53e2K92opLE0LL93t5r/AORCz7nhNh8atautC1W8fX/BbPZrEUeOz1IRpufad4Kbjntt79eKPDvxq1rVvEFpYz6/4LmjmYgpa2epLIflJ+UugXt3r3aih4mhZ/u9/Nf/ACIWfc+dv+F96/8A9DJ4D/8AAHVf/jdafiL41a1pOuz2cOv+C4kjWMhLmz1JpBuRW5KIV78Y7YzzXu1FP6zh739l+K/+RDlfc8JvfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YosPjVrV1oWq3j6/wCC2ezWIo8dnqQjTc+07wU3HPbb368V7tRS+s0LW9n+K7/4Qs+54T4d+NWtat4gtLGfX/Bc0czEFLWz1JZD8pPyl0C9u9Zn/C+9f/6GTwH/AOAOq/8Axuvomin9Zw97+y/Ff/IhyvueE+IvjVrWk67PZw6/4LiSNYyEubPUmkG5FbkohXvxjtjPNF78atat/D+l3ya/4LWS7aYO8lnqRjfYwA2AJuGM87vbFe7UUliaFl+7281/8iFn3PCbD41a1daFqt4+v+C2ezWIo8dnqQjTc+07wU3HPbb368UeHfjVrWreILSxn1/wXNHMxBS1s9SWQ/KT8pdAvbvXu1FDxNCz/d7+a/8AkQs+587f8L71/wD6GTwH/wCAOq//AButPxF8ata0nXZ7OHX/AAXEkaxkJc2epNINyK3JRCvfjHbGea92op/WcPe/svxX/wAiHK+54Te/GrWrfw/pd8mv+C1ku2mDvJZ6kY32MANgCbhjPO72xRYfGrWrrQtVvH1/wWz2axFHjs9SEabn2neCm457be/XivdqKX1mha3s/wAV3/whZ9zwnw78ata1bxBaWM+v+C5o5mIKWtnqSyH5SflLoF7d6zP+F96//wBDJ4D/APAHVf8A43X0TRT+s4e9/Zfiv/kQ5X3PCfEXxq1rSddns4df8FxJGsZCXNnqTSDcityUQr34x2xnmi9+NWtW/h/S75Nf8FrJdtMHeSz1IxvsYAbAE3DGed3tivdqKSxNCy/d7ea/+RCz7nhNh8atautC1W8fX/BbPZrEUeOz1IRpufad4Kbjntt79eKPDvxq1rVvEFpYz6/4LmjmYgpa2epLIflJ+UugXt3r3aih4mhZ/u9/Nf8AyIWfc+dv+F96/wD9DJ4D/wDAHVf/AI3Wn4i+NWtaTrs9nDr/AILiSNYyEubPUmkG5FbkohXvxjtjPNe7UU/rOHvf2X4r/wCRDlfc8JvfjVrVv4f0u+TX/BayXbTB3ks9SMb7GAGwBNwxnnd7YosPjVrV1oWq3j6/4LZ7NYijx2epCNNz7TvBTcc9tvfrxXu1FL6zQtb2f4rv/hCz7nhPh341a1q3iC0sZ9f8FzRzMQUtbPUlkPyk/KXQL271mf8AC+9f/wChk8B/+AOq/wDxuvomin9Zw97+y/Ff/IhyvueE+IvjVrWk67PZw6/4LiSNYyEubPUmkG5FbkohXvxjtjPNF78atat/D+l3ya/4LWS7aYO8lnqRjfYwA2AJuGM87vbFe7UUliaFl+7281/8iFn3PCbD41a1daFqt4+v+C2ezWIo8dnqQjTc+07wU3HPbb368UeHfjVrWreILSxn1/wXNHMxBS1s9SWQ/KT8pdAvbvXu1FDxNCz/AHe/mv8A5ELPufO3/C+9f/6GTwH/AOAOq/8AxutPxF8ata0nXZ7OHX/BcSRrGQlzZ6k0g3IrclEK9+MdsZ5r3ain9Zw97+y/Ff8AyIcr7nhN78atat/D+l3ya/4LWS7aYO8lnqRjfYwA2AJuGM87vbFFh8atautC1W8fX/BbPZrEUeOz1IRpufad4Kbjntt79eK92opfWaFrez/Fd/8ACFn3PCfDvxq1rVvEFpYz6/4LmjmYgpa2epLIflJ+UugXt3rM/wCF96//ANDJ4D/8AdV/+N19E0U/rOHvf2X4r/5EOV9ytp1w13pdpcu0btNCkhaIEISVByoPOPTPNWaKK89tN3RS0WoUUUUhlMP5eZMZ2qWx68f5/P354Y/EPxIunG+bwxpXkLpSaqca1IW8lskDH2XG/jpnHvXeQ/60f5/z1/X355xJ9V/4WPJcHw3qIsGtFtBema18vIkZt+3zt+3BH8O72pL4l2/4D/Ww38L/AK6r9LmtqnibTNHultrprqWcqHMVnZTXTIpzhnESMUBwcFsA4PoapeIPGmn6FplhfLFc38N9NFHE9lazTrtd1UtujRh0bIU4LEYHNMl/tTQdc1K5tdHu9Zt9TljmX7LNCr27LGsZVhLIg2/IGBUk5ZuOmak+ganaeCIbeCBLq+i1JdSe1jlCg/6X9oaJGbA4BKrnAOBnHZrp6r/ghpc37LXrO/0ye/hiv44ICd4uNOuIJDgZO2ORA7dewOeg5pmleJdN1m4e3tHuY7iNd5gvLOa1kK5xuCSqrMueNwBGeM1k6jeeJ9W8O6h9j0W40mYGMQRteQ/apk3jzQNpaONimQrbzyeduM1k+HdK1Kz8c2t+vh/WrXTnsp7ZpNT1v7bLE5aNgWVppNqnYQPLZiT94KOaa+Kz/rQTWlzobO/1iPxrPpmo3FjPZyWrXVv5Fq8UkYEgUK7GRg/B6hV+ldBXFNqOt/8ACbDUB4M1k2y2Ztd4nseT5gbdj7RnbgfX2rtaS+FfP83+lgfxP+ui/Uxtb8W6N4cura31i4mgkusiDbayyCVgCdisqkFzg4TO49galsvEWnahYXF3bPcbbYZmhltJY54+M8wsok5HI+XntmsXxpLdQa94WnsLFb+aG+mYW+9UZx9mlB2M3AbBOMkA9CRnIr39n4h1GHVdZsLG60y7uLe2tY7Izwi5aOOV2kIdWaNXZZGCfMcHkkZ4Og1ZnQ6V4l03Wbh7e0e5juI13mC8s5rWQrnG4JKqsy543AEZ4zVCTx94ehuLq3lubpZ7N9txCdPuN8Q/56FdmfL5/wBZ9z/arC8O6VqVn45tb9fD+tWunPZT2zSanrf22WJy0bAsrTSbVOwgeWzEn7wUc1f0e71eDxhq09x4S1WK31CeEJctNZlUVECFmAnLYzk4AJx2zxR2F3NvU/FGm6RMkd59tIZBJ5tvp9xPEqn+JpI0ZVHGckjA5qnq3j/w3olzDDqN9IhuLf7VC8dpNLHJF1Lq6IVIAIJOflBBOBzWJ4g0vW73X78S6brN/FJtGnS2WtmxtYF2DcJgkiuW37iTsk4KgY5FUbXTfED6dpNlL4ZvoWh8Mz6bLLJcWpRJysYUcTEkHyjyAfvrnHzbZu+Vv+tm/wBEvmWoq6T/AK2Oy0zxfoWr2d5dWOoI1vYr5k8sqNEqxkFhIC4G6MgEhxlTg4JxSab4u0jVb2K0tXu0nmVmiS5sJ7fzVUAllMiKGHI5HHNYFla6skt81x4XmuI7zRbWA29zcW4RnjMqvC5Dt1WQHIDKRnnPFWPDthqVr4gQ2unavpmmLHILlNV1P7YJn+Xy/JHnSFAPmz9zggbT/Do0lJpGe8UzfPiPSl8UL4dN1/xNWtftYt/Lb/Vbtu7djb17Zz7Vp1nNe6mPEiWS6Tu0xrcyNqX2lfllzgR+V945HO7pxWjRJJJW/r/IfUzdavrq1S0g08Qi6vLgQRvOpaOP5GcsQCCflQ4GRkkc0/TbyWaS4tLx4nu7RlEjwoUVwy5DBSSV7jGT061Hr8CXGmiKXTLjUlMiny7WVY5EI5DqzOmCCOoYHmoPDuiJpIu5ltxbNdyBjF5hkYBRgF3OSznkkknqBk4ycfe5zuSpfVm38Xy1d153Wl9LW63ubVc9c6lrdiy3l5HZrZvdLAtmqsZ9rSBFfzN20nkMU29MjdxUnhrxG/iOTVnWy+z2ljfyWUE5l3fafL4dwMDA3ZUcnO09Kow3Oszakt1qHhu+eZZdsCfaLYQW6E4L58wszbcnO3P8IA5JdVShJRYsJFScm7WXdpP0V2vm+n4PR13XBp8i28d5YWT7A8l1fviOEE4Xjcu4sQ2BuH3Tz0Bs3epHSvD7X166XUkcYI+zJtE7nhVRSx5YkADJ5PWs7UtOntPFkevW2nSakTZ/ZTFHIivEQxYMu8qvO5gTnIwMA5NJJ4emuNCsVnuLq2ubGV7qOOyMRw53YjHmKUIUNgZAHAPFZ3lZ9/6/TU3VPD8lO7VtL7X6380ul9tb201u+GNSvNW8PxXWppCl15s0UiwAhAUldOMkn+H1rWrnvBNlf6f4feDVY7qOb7XPIFumhLbXkLgjyuOd2TnncWxxipPDXiN/EcmrOtl9ntLG/ksoJzLu+0+Xw7gYGBuyo5Odp6VvTjKUObsjlxihDETULW5na23W1vkV5rnxLYyWH2y90l2urtITbxWcgJUks2JDL1CKxzt6jpV+8u9SuNSls9Ie1iNtGkkr3MTSByxbCAKylT8uSxz94cHmpJrSa48SWs7xj7LaQOUfI5lcgdOvCg/991BdLeadrE95ZafNfreRRoUikjTynTdy29h8pDDpkjb0Oaxs1Hrv8/6udHPGdtI81vJK99u3w9/zsSNqdzd+HYtR0uOEGWHzT9oY/uxtz0A+Y54xkfWrmnXD3el2tzIFDzQpIwXoCVBOKzDb3Wj+E7fT4LKfUp1txAwtjGvzbeW/eOo259yeas+H3uTotvDe6fcWMtvGkRSdo23YUDI2Owx9cH2qot8zUvL063M6sIezcoWtzaaq9vS9/wADTooorQ4QooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigCtD/rR/n/AD1/X35s1TCeblM43KRnrjI/+v8A5zz5qPhZ4bOqyaCuk6Us6+H44xfDTY9wm3spnx13nGc5z70L4kv62b/Qf2b/ANbpfqerUVymvjWf+Ey0m20zXJrO2voZlmhNvFIqbAp3oSm4P82PmLL/ALJpmtXGo6bp1npKaprN/q0rO8cml21oLiWJTyX84eSuNyAthcnGBzilcDrqqWGqWeqLcGxm80W1w9tL8pXbIhwy8gZwe44rB8I3+qaxo+q2uqSXlrdWl01ok06QC5UeUjh2Ee6IuPM6gbTgfL1FWfCct9LZ6nbahqU9/Ja38tvHczJEshUBSMiNFXIz/dp9beV/y/zF0+dvz/yOgorzixu/E9toCa5d+JJboQaobT7JJawBJ4ReGDMhVFbzMcgqVUYHynnOh4k1XU5dYuIvD0viKU2KhbhdNgsTbpJt3gObnDsdrKSI24BHIJpN2Vx2d7G1rGm+EtV1y0ttfs9FvdV8svaRXsUUk+wHJKBgWwDzxWzcXENpbS3N3NHBBCheSWRgqooGSxJ4AA71wWq3eo67beAbq2vItNvNRfzZJhGHMe+ykZhGrZG7rjdkDqQcYO/oVxqFr4i1DQtR1FtVW3tobqK6ljjSYCRpFKSCNVTgx5BCjIJz0yacWrxEtdTX0zVtO1qzF5o2oWuoWxYqJ7WZZUJHUblJGat1z3hZth19sE41aY4HU/KlclpOu+M9QtbXWbbTvEFx9qCzC0caWLIIxB2qRKJxheAxJOeSv8NJa280n96B6X8m/wAD06iiuS8XPrI8ReHrXR9dfTI9QuJbedPIikBUQySbk3KT5nyADkrjJKmgDraK46TV7nQZNV0zVNVvL0QWkE1verbxNdF5pJI1j2IgRm3INvyAHJz0zUPhfVddXxUNK1dNY8iaykuUbWUsvNDI8a4Q2p27cSchhnI64prV2B3R29FcVbW3iK+8Ra9YxeKrmG3tJY2t2NnbtIpdA+wnYAYxnGMbz/fFM8R6rqkurTxaBL4hmexRVuRpkFj9mSXbuw5ucO3DKSI24BHIJqbq1x2uzuKK801PVvEmoW2mahp2vyaYLjw5Jqk1tHawyR+aixEBS6lgCZCGG48Djaea1tBvPEjx6xaSanDqN/8A2bb3lm9xAsUcc0yy/JhBnyw0a4zlsE5Y8U9r+X+bX6MLbef/AAP8zsbe2gtIRDaQxwRAkhI0CqCSSTgepJP1NSVxXh271mLXra013UNYheaORlttUgsiLkqFz5T2x+ULnO1/mIPsa6I2OqnxQt8NYxpQtfKOl/ZV5l3Z83zfvdONvTvVWu9WSadMnnitoHmuZUhiQZeSRgqqPUk9KfVLVreCfTpmnhjkaKN3jLqCUbaeRnofespycYto0ppSklLYto6SxrJGyujAMrKcgg9CDVPSrrSZoZIdEnspI4HIkjtHQrGxJJBC9CTk/XNZV4zL8NSVyP8AiWqGKkghdg3HI5HGat3AWHxZpiQBU3WlwrKoxlFaLH4Anj0z70Sk1Kx0rDxs9dfet/26r/ibFVb3U7DTfL/tG9t7TzW2R+fKqb29Bk8n2q1WPAizeKNUW4QP/osCAMMgxkyZ46cnOfXAz0FOTa2MaUIyu5bJX/FL9TTubmCztnuLuaOCCMZeWVwqqPUk8Ci2ure9tkuLOeO4gkGUlicMrD1BHBrmdNke403wiZyZEf5gxOdzCByh9+Mnn0B61o6dFHPqOvWssaSWxuVDRMoKktDGWBHvnJHvnvUKbb0/rS50VMNGmmm9Vr8ubl/4O5q291b3aM9pPFOqsUZonDAMOoOO49KlrH8PqsbaoiKFVb9wFAwANicVsVcXeKZzVoKE3FBRRRVGQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAFaH/Wj/P+ev6+/OYvhDSl8RHXA2o/byeWOq3RjIyTt8rzNm3JJ27dvtWgDIAxhVXkCkorttBOOMnBwPfB69/4uKtfiTqF94WvtQtdCtV1DTZ9l3Yz6iyKIycJJHIITvBPHKryrDtydQ6HSar4N0nWtUTUb5tSFzGpVGttWurcICMHasciqM4GcDnHNTaj4W0vVYbNLxbvfZKUgnhv54Z1UgAgyo4dgcDOWOSATkim33iex0loYNSE5vGjV5YLC0nvPKz3byoyVXIIDMFzg+hpmo+MtC0qKylvLxvKv1LWssNvJMk2BnaGRSNxH3VzuboAaNBq9xLDwZoml2t/baZBc2cWoENcLb308eWAxvUhwUc4G5lwzdSTTdK8F6Rot49zp76oJJCzOJtYu5kdmGCxR5SpbHcjNaGj63Ya7avcaZMzrFIYpUlieKSJwASrxuAyHBBwQDgg96v0f18hHOf8IHoX9gnRtuo/Yjc/ain9rXW/zM7s+Z5m/G75sZxu+bGeakvfBGg6jeyXN5azyNMqrPF9tmWK52jaPNiD7JTgAZcEkAelU9K+IOnapqd1ZCw1iJ4bsWqM2j3m1sojbnYwgR8vjDEYADdCK0dU8W6Po2qR6fqU80NzNGXhH2SZlm/2UcKVd/8AYUluelD2u+o9U7FOT4eeG5rPT7Wa1upYNMQpZpJqNw3kZyAy5k4YAkK33lGApAAq/a+F9MstLubG2W7RLo7p5vt05uJD6mcv5uQAADu4HA4pD4p0waPHqafbZreRzGFg0+4llVgSCrRKhkUggggqMd6sWeu2GoaVLqFnJLNDDu81Ft5PNRlGShi27w/+xt3cjjmh9bi7FHSPBuk6FfNd6c+piVmZnE+r3c6OzDBZkkkZSfcjNIfBGgm7kmFrMqSuZJLVLyZbWRicktbh/KJJ5OV5PJqTwzqi6np95drfz3qC7lCiawe1kt1ByIjGyhiVB6kZOaq2fxC8N39xDFb3s4M0ogDy2M8aJKWKiJ3ZAschIwEYhjkccjJ1QdGdLXI+NPD+p6/qGkC10/T7yys5WmmFzqEttJuKMg2GOJiCu7cGDAgjjHWtjVfE2naNcLDfrfAld5kh064mjRfVpEQqg4OdxGBzWZ4g8WDQ9S0O5e9szouol42KwPJLI3lNIhiZGO7O0AKEYnPBo8wNBfCWjjRrjTJLeae3uSGma4upZZnYEFWMzsZMqQNp3ZXAxjFQ2HgjQtN1eHVbWG6OoQqyC6m1C4lkkRsZR2dyZFGAQr5CnkAGrVhrlrr9jdHRLnZcxAoVu7WSN4HI+UyQvscDvg7cjoe9Q+F9Q1G+tb5NZktZrmzvZLYy2sDQo4UAg7GdyDz/AHjR18/6X6h08v6/yIbXwPo9nqzalbyasty8olctrV4yyMOm5DKVYDpggjHGMU/UfBeharqE15fWszvcBRcRLdzJDcYGB5sKuI5OAB8yngAdhSxeM9Em1D7Ik9wCZfJW4axnW2d87dqzlPKY7uAAxyeOtbtKysO7TObt/AHh+1Nv5UV9ttrR7KJH1S6dFgfhk2tIQR06jjauMbVxKPBOhCN0a3uZEktEspFlvp5BJErFlDBnO4gk4Y5bBIzg4qbW/FujeHLq2t9YuJoJLrIg22ssglYAnYrKpBc4OEzuPYGpbLxFp2oWFxd2z3G22GZoZbSWOePjPMLKJORyPl57Zp76/wBf1q/vEM07wxpel6gb22S5kuSpRZLq9muTGpIJVPMdtgOBkLgHA9Kmbw/pj+JE19rbOpx25tVn8xuIidxXbnb1HXGaj0rxLpus3D29o9zHcRrvMF5ZzWshXONwSVVZlzxuAIzxmqEnj7w9DcXVvLc3Sz2b7biE6fcb4h/z0K7M+Xz/AKz7n+1TUnHZgdJVTUdMt9VtxBdtcCMHOILmSEnjGCUYEjnoeKp6n4o03SJkjvPtpDIJPNt9PuJ4lU/xNJGjKo4zkkYHNU9W8f8AhvRLmGHUb6RDcW/2qF47SaWOSLqXV0QqQAQSc/KCCcDmpaTVmVCUoyUoOzNbTtItNLsza2vntAeNlxcyT4GMYHmMcDHYcUmn6NZ6Y7vaLKXcBS01xJMVUdFUux2r7DAqnpni/QtXs7y6sdQRrexXzJ5ZUaJVjILCQFwN0ZAJDjKnBwTik03xdpGq3sVpavdpPMrNElzYT2/mqoBLKZEUMORyOOaOVJ7FyrVXe8nrvrv69yEeG5JviK3iS8kjeO308WdlEMkxlnLSucjAJwgGOwNaeoaPZ6oyNdrNlARmK4ki3A9VbYw3L7HIqI+I9KXxQvh03X/E1a1+1i38tv8AVbtu7djb17Zz7Vp1dS8klNaW09P6uRCcoS5ouz/4H+RWurC2vLP7LMjCIY2+U7RsmOhVlIKkeoIpsOl2tvpzWUIlSJwQzCd/MJPUmTO/d/tZz71BrV9dWqWkGniEXV5cCCN51LRx/IzliAQT8qHAyMkjmn6beSzSXFpePE93aMokeFCiuGXIYKSSvcYyenWsvdb2OjlqqlzKWm9r/K/36dyPTfD1hpNy89mbzfJkuJr+eZWJxklXcjPA5xmtOiueudS1uxZby8js1s3ulgWzVWM+1pAiv5m7aTyGKbemRu4o0grJCiquJleUrvzer8v60Ohormdc1+ax8QJYDWdI0qM2yyq2oxljKxZlwv71OmB69a2NQ1EaTosl7dfv3jQfLCuPOkOAqqCTyzEADJ6jmkqiab7BLDVEod5bb/5W+5l6isnwxqV5q3h+K61NIUuvNmikWAEICkrpxkk/w+ta1WndXMqtOVKpKnLdNr7gormprnxLYyWH2y90l2urtITbxWcgJUks2JDL1CKxzt6jpV+8u9SuNSls9Ie1iNtGkkr3MTSByxbCAKylT8uSxz94cHmoU7q9v63N3hmre8rd9fTt37GtRWM+sy3em6a+mqkc+p/6szLuWEbC5LKCC2MYwCMkjnFWdKvJ5/tNtfGJrqzlEcjwoVR8qGDAEkjhumTyDzVKSbsZyw84xbfTp13tf79DQoooqjAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAKYdkDOkbSsqlgiEZY46DJxk+5A5685PmNx4Z8R3Xhi1lsdCuLLU43ezvLa4mt83Vq8pkDBkkYZjYhhkg4MgA+bn1GH/Wj/P+ev6+/Nmjrf8Ar+ugdLHMy/2poOualc2uj3es2+pyxzL9lmhV7dljWMqwlkQbfkDAqScs3HTOWujapp0/hiNtIlv/ACdTub67kt5IfKtPOE3A8x1YhTOPuqThScZwD3VFC0A5C10d9a1HxZYeINDuYtL1KVAryzRhbmMRJGceXIXXJQnkKcEd8gHhz4VeDPCWsLqnh7Rvsl6qMgl+1TSYU9RhnI/SuvorSNWpGLjGTSe/mDSe5yKy6/outazHY+HJNRXULxbm3uheRRQIDDGhEhY+YpBjJ+WN+CPcBPEM+qx+MdKurPwvqWpW1jHNvntprVQTIqgBRJMjcYOcge2a6+is+iXYbd3c5LxVDq19/ZckWnavPp7I7Xljpl/HaXSyEKY90nmplV+cEJIOdv3hVbwdb6toNlrwvPD+oAveC6tov7RS7knRoo12iWWQMXBQ53kAZwrMOa7aihaXt1A5Dwpd6vDfajHqHhbVLGO7vJbpJ5prRkVSq4UhJ2bcSuOAR71kraa0vw7ltE8K363z6w1wbUTWgcobv7R5m7ztn3fl+9nd2xzXotFC0t5WX3C7+tzz/XdN1y/1y+M+la1eRyqn9ntZ659htrddg3LMI5Vctv3EnZKMFcdxVdLXxJb6X4HEHhWeWbRY83kc11bjZtt3h2owkOXJIK/w4PzMpzj0iihaKy/r+rjepy2njU/7T1TxJc6HcwPLaw20GmLJAbmQRtI25mEnlgkyEAb+ADk5OBD4VuNWF7qVvqPhnVNOivryW4S5mmtGRFZQAD5czNu47KR7119FHX5WE9fz/Q8v0/wbPZ2MOi3mg69etAoiF0viidbGVV4VmT7RvTIGSgiYA8DI5r1CiijUOtzkfGkt1Br3haewsVv5ob6Zhb71RnH2aUHYzcBsE4yQD0JGcivf2fiHUYdV1mwsbrTLu4t7a1jsjPCLlo45XaQh1Zo1dlkYJ8xweSRnjo9T8NaFrdzDcazounahPAMRS3dokrRjOflLAkc88Vp0dB3sefeHdK1Kz8c2t+vh/WrXTnsp7ZpNT1v7bLE5aNgWVppNqnYQPLZiT94KOav6Pd6vB4w1ae48JarFb6hPCEuWmsyqKiBCzATlsZycAE47Z4rsqKO3kI4HxBpet3uv34l03Wb+KTaNOlstbNjawLsG4TBJFctv3EnZJwVAxyKo2um+IH07SbKXwzfQtD4Zn02WWS4tSiTlYwo4mJIPlHkA/fXOPm2+mUVPL7rj/WzX6lc1ndHC2VrqyS3zXHhea4jvNFtYDb3NxbhGeMyq8LkO3VZAcgMpGec8VY8O2GpWviBDa6dq+maYscguU1XU/tgmf5fL8kedIUA+bP3OCBtP8PZUVbbcnLuQtEkZzXupjxIlkuk7tMa3Mjal9pX5Zc4EflfeORzu6cVo0UUN36DMzX4EuNNEUumXGpKZFPl2sqxyIRyHVmdMEEdQwPNQeHdETSRdzLbi2a7kDGLzDIwCjALuclnPJJJPUDJxk7VFZ8q5uY6FiJqi6K2fr+V7dO1zC8NeI38Ryas62X2e0sb+SygnMu77T5fDuBgYG7Kjk52npVGG51mbUlutQ8N3zzLLtgT7RbCC3QnBfPmFmbbk525/hAHJPT29tBaQiG0hjgiBJCRoFUEkknA9SSfqakqqkVKScdEhUa3sub3U7976LsrNej/4czL+/wBQtZmSDRZb+Jl+R4J4xz3DhyuB7jd9Kzm8Nyv4b0uCe6uYbrTQJlFiYzmQKRtAlUqQNxAyB2PHbpKKhwT3LhiZU0lBJfro1rdtdWc94Jsr/T/D7warHdRzfa55At00Jba8hcEeVxzuyc87i2OMVJ4a8Rv4jk1Z1svs9pY38llBOZd32ny+HcDAwN2VHJztPSt2o7e2gtIRDaQxwRAkhI0CqCSSTgepJP1Naw5Yx5dzLEVZV6sqrSTbvpsUZrSa48SWs7xj7LaQOUfI5lcgdOvCg/8AfdQXS3mnaxPeWWnzX63kUaFIpI08p03ctvYfKQw6ZI29Dmtmis+UuNdqyaTVrW12vfvffUwF0m403SdHMKNeT6WAHiiKqZgYyrbdxAHJBGSOBjNXNHt7hZb6+u4Wt5L2YSCBipaJVRUAYqSCflJ4JHIHbNadFCgk7oc8TOcWpbvr8727b67X+QUUUVZzBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBWh/wBaP8/56/r782arQ8yKfbP6f/X/AJ++bNABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAH/9k=)

**#train the model**

**history = model.fit(X\_train, y\_train, batch\_size=128, epochs=6, verbose=1, validation\_split=0.2)**

**score = model.evaluate(X\_test, y\_test, verbose=1)**

**print("Test Score:", score[0])**

**print("Test Accuracy:", score[1])**

**#plotting the loss and accuracy differences for training and test sets**

**plt.title('model accuracy')**

**plt.ylabel('accuracy')**

**plt.xlabel('epoch')**

**plt.legend(['train','test'], loc='upper left')**

**plt.show()**

**plt.plot(history.history['loss'])**

**plt.plot(history.history['val\_loss'])**

**plt.title('model loss')**

**plt.ylabel('loss')**

**plt.xlabel('epoch')**

**plt.legend(['train','test'], loc='upper left')**

**plt.show()**

**OUTPUT:**

![A close up of a map

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDcRXhpZgAATU0AKgAAAAgABAE7AAIAAAAGAAAISodpAAQAAAABAAAIUJydAAEAAAAMAAAQyOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEhhcnNoAAAFkAMAAgAAABQAABCekAQAAgAAABQAABCykpEAAgAAAAM2NwAAkpIAAgAAAAM2NwAA6hwABwAACAwAAAiSAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAxOToxMDowNyAxMjozNjoyOAAyMDE5OjEwOjA3IDEyOjM2OjI4AAAASABhAHIAcwBoAAAA/+ELGGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTAtMDdUMTI6MzY6MjguNjY2PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkhhcnNoPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIASIBzQMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APpGiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoqrfanaaabYXsvlG6nW3h+UndI2cLwOOh5PFWqACiqul6naazpdtqWmy+daXUYlhk2ldynocEAj8RVqgAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAorM1rWv7HFkq2F1fzXtx9nhhtTGGLeW8hJMjqoG2Nu/pVX/hINS/6FDWv+/wBZf/JFAG7RWF/wkGpf9ChrX/f6y/8Akij/AISDUv8AoUNa/wC/1l/8kUAbtFYX/CQal/0KGtf9/rL/AOSKP+Eg1L/oUNa/7/WX/wAkUAbtFYX/AAkGpf8AQoa1/wB/rL/5Io/4SDUv+hQ1r/v9Zf8AyRQBu0Vhf8JBqX/Qoa1/3+sv/kij/hINS/6FDWv+/wBZf/JFAG7RWF/wkGpf9ChrX/f6y/8Akij/AISDUv8AoUNa/wC/1l/8kUAbtFYX/CQal/0KGtf9/rL/AOSKP+Eg1L/oUNa/7/WX/wAkUAbtcv8AEaa5g8EzmxvJ7Kd7uzjE9u+10D3USnB+hI9DnBq3/wAJBqX/AEKGtf8Af6y/+SKP+Eg1L/oUNa/7/WX/AMkUAcFrEN5pK+MJbTW9Y26E9tcWEcmoSuI2ZFZwxZiZFYj7rlgMnAFTalq9yniO/B1e9j8Rx63bw6fpS3DiOWzLRBj5GdroUMjNIQSpB+YbcV2//CQal/0KGtf9/rL/AOSKP+Eg1L/oUNa/7/WX/wAkUAebTX8V3r2if2hrF4/iD/hKWS6017hmSGFXlEWISdqLsEZVwAW3dTmrfhTVdbu/FtiL/VbKDUjeTjULCbW5nleICTCLZGIJGFwhV1bkLyzbq77/AISDUv8AoUNa/wC/1l/8kUf8JBqX/Qoa1/3+sv8A5IoAxfBV2LD4H6XeNHPKLfRhKY7YZlfbGThP9o44964rTfFGrN/b0Xhy7adG0MXcIt9Xl1ZopBJtZleRBiQIxJjUsMqvHOD6f/wkGpf9ChrX/f6y/wDkij/hINS/6FDWv+/1l/8AJFAHI3+o6NDotkdA8TXlxo8upRx6zfpqstw9tEYpCMyliYQ0giVsFdoY/d61aTxBpOieJdEni16STw7cabeRw3M189xHPOJ4Sq72Zt748wLySQCBmuk/4SDUv+hQ1r/v9Zf/ACRR/wAJBqX/AEKGtf8Af6y/+SKAOF8Fve+IbrwqupatqjRHw0l3LGl7LH503mqA7lWBY4zkE85wciqeieLbu68caf8A2JJJi/F4k1lcazLdyK6Rs6CWBl2Wzbk4VW6E8HGR6N/wkGpf9ChrX/f6y/8Akij/AISDUv8AoUNa/wC/1l/8kUAcdozpq/gmZtD8R3114tFilzc28uoyF47lCrmN4N2IQXXYVCqCpIwRXQ+BNXk8UR6j4mWWcaffyrFp8EhICQxDaW2ngM0hkyccgJ6CrF/q+rXlhLbw+HPENk8i4Fxby2HmR+675mXP1Bpmk3tzouj2umWHg3WktrSJYowZ7MnAGMk/aOT3J7mgDqKKwv8AhINS/wChQ1r/AL/WX/yRR/wkGpf9ChrX/f6y/wDkigDdorC/4SDUv+hQ1r/v9Zf/ACRR/wAJBqX/AEKGtf8Af6y/+SKAN2isL/hINS/6FDWv+/1l/wDJFH/CQal/0KGtf9/rL/5IoA3aKwv+Eg1L/oUNa/7/AFl/8kUf8JBqX/Qoa1/3+sv/AJIoA3aKwv8AhINS/wChQ1r/AL/WX/yRR/wkGpf9ChrX/f6y/wDkigDdorC/4SDUv+hQ1r/v9Zf/ACRR/wAJBqX/AEKGtf8Af6y/+SKAN2isL/hINS/6FDWv+/1l/wDJFH/CQal/0KGtf9/rL/5IoA3aKxbHxFJda1Hpl5ouoabNNbyXEb3LQMrrG0asP3crkHMq9QO9bVABRRRQAUUUUAFFFFABRRRQAUUUUAFeY3UtnD46mlneGPX01tGR3IEi6eLZSxz1EO3fn+Hfn+KvTqKAPKPEBt5fiV9oWe0a4XUNOEFu6ZvJoyVLPbSZ+WEBiXG1gwSXJXOa2PClz4d1H4gX1x4Yk022itYJraaG2kQS30vmIXmdAclUKlVduSZH7YLd/RQBzOvR33/CXeGXa5tzY/2i4WAW7CUP9jueTJvwR142fjW+BJLNKBM0aowUBQOeAe4PrWT4g/5Dnhf/ALCj/wDpFc1cvLa8uoriPTr77DN5wPm+SJONg4waa1Ym7It+TJ/z8y/kn/xNHkyf8/Mv5J/8TWJ/Y3iT/oav/KdH/jR/Y3iT/oav/KdH/jWnJH+Zfj/kZ+0l/I/w/wAzb8mT/n5l/JP/AImmlcSiI3jiRlLBPkyQMZOMdOR+dM022vLW1Meo3326bcT5vkiPj0wK4L4madePq0N/p66yLj+x7u3t5tLa4JjnMkLRgiLgZAc5YbTtGegrNqzNE7o9C8mT/n5l/JP/AImmQ7bhC8F80qq7IShQgMpKsOB1BBBHYivPvE6a1pviCzg0JNW8u0e0Mbg3l0LpXuD5wd/M8pQqE58wMcEbcYGEl0/XtQnlS5n1yKOOHWpI/JnmizILxfs3KkZ/d52L0K9ARSGehyCSEK/nu43qpVguDkgdgPWrFZ1pJcTaDYy3qsty8cDTKy7SHJXcCO3OeK0aACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKCQBk8CgAorLvPE+h6fkXmr2cTD+EzqW/IHNUf+E10+b/kG2mpaj72tk5X/vpgB+taKlN6pGbq007NnRUVzv8AbXiG5/48vDJhU9HvrxI8fVU3Gj7L4vuv9dqOl6ev/TtbPM35uwH6U/ZtbtL5/wCVxe1T2Tfy/wA7HRUyWaK3jLzyJEg6s7AD8zWB/wAIrc3HOpeJNWuD3WGVbdT+CKD+tPi8EeHY5BJJpkdzJ/fu2acn/vsmjlpreX3L/Owc1R7R+9/5XLtv4j0a71BbG01O1uLlwSIopQ54GT0rSqG2s7azTZaW0MC/3Yowo/SpqiXLf3S481veMK7/AOSh6R/2C77/ANG2lbtYV3/yUPSP+wXff+jbSt2pKCiiigAooooAKKKKACiiigAooooAKKK4iTXNVHiKS6F8wsotbj0r+z/KTayNEpMm7bv3733fe27RjGeaAO3orgda8Qa1beNmitriddNt72ytpGjjha2j80qGSYkeb5h3rt2fKN6FsDNT+G9c1W61bS5r2+a4t9agu5hamJFFoY5FCBSFDEbWIbcWy2CMdKANrxB/yHPC/wD2FH/9IrmteH/XXH/XQf8AoC1z+vXUx8XeGbU2FwsI1F3F4Wj8pj9juflA378/8Bxx1rZmtfti3EXnzwfvgd8D7W+4vGaUm0rpDik3Zl2isn+wP+otqn/gT/8AWo/sD/qLap/4E/8A1qw9pV/k/FG/s6X8/wCBrVC15brfx2TSqLmSJpUi7silQzfQF1H40lna/Y4PK8+ef5s7533N9M1w3xD8J3mt6tbXmm6Wl1P/AGdcWq3K+SHt5GkhdG3ScgbVlXKhiC3Tk1vFtq7RhJJOyPQKhgvLe5nuYYJVeS1kEUyjrGxRXAP/AAF1P415lL4LlXw6YrXQ9Wkdbw3EVjeR6a8IfytmWhRli2HPYh9wz3Oej8J6DLpPiXWLu80K2tri/wDKmF9bLHsA8iFHhBz5n+sjZuV2kYOc0xHVXX+pX/ron/oYqaobr/Ur/wBdE/8AQxU1ABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFMmmit4mluJEijXq7sFA/E1hS+NdJMjRaabjVplODHp8Jlx/wACHyj86uMJS2REpxj8TOgornPtnivUf+PXTrPSYj/y0vZfOkx6hE4B+rUf8InJe86/rd/qAPWGN/s8J/4DHgn8Sar2aXxP9f8AgfiT7Rv4Y/p/wfwL+o+JtG0l/LvtRgjlzgQq2+Q/8AXLfpVD/hI9TvuNE8PXUinpPfsLZPrg5cj/AIDWrp2iaZpCbdMsLe145McYDH6nqfxq9RzU1sr+v/A/zDlqPd29P83/AJHOf2T4j1D/AJCmupZRnrDpcOD/AN/HyfyApw8DaLIQ1/Hc6i/969u5Jc/gTj9K6Gij2s+jt6aB7GHVX9dSlZ6LpenY+wada2xHeKFVP5gVdoorNtvVmiSSsgooopDCiiigAooooAwrv/koekf9gu+/9G2lbtYV3/yUPSP+wXff+jbSt2gAooooAKKKKACiiigAooooAKKKKACsibwvpE+sNqklvJ9rY7iyXEirv2eX5mwMF37DtD43AcA1r0UAY83hTR7jVE1Ca2ka4Vo3P+kybJHjxsd03bXZcDDMCRgc8CpLDw1pWmanNf2Vs0dxLvBJmdlQO29wiMSqBmAYhQMkAnNalFAGF4g/5Dnhf/sKP/6RXNa5ikWR2hkVQ5yQyZ5xj1HpWR4g/wCQ54X/AOwo/wD6RXNbtAEO25/56xf9+j/8VRtuf+esX/fo/wDxVTUUAQ7bn/nrF/36P/xVG25/56xf9+j/APFVNRQBDtuf+esX/fo//FUbbn/nrF/36P8A8VU1FAEBimfaJZUKhg2FQgnByO5qeiigAooooAKKKKACiiigAooooAKKKKACiqGpa7pejru1O/gts9Fdxub6L1P4Vl/8JPe3/GgaFd3SnpcXf+jRY9Ru+Y/gtaRpykr20M5VIRdm9To6q3+qWOlw+bqV5Bax9mmkC5+metY39keIdS/5C2uLZRnrBpce0/8Af18t+QFWrDwnounzefHZLNc9Tc3JM0hPrubJH4U+WC3d/T+v8yeacvhVvX+v8ir/AMJcb3jw9pF9qmekxTyIT/wN8Z/AGj7F4p1Hm81O10mI/wDLKxi82THoZH4B+i10dFHtEvhX6/8AA/Afs2/il+n/AAfxOfh8FaOJVm1CObVZx0l1CYzf+On5R+AreiijgjWOGNY0UYCouAPwp1FTKcpfEyowjH4UFFFFQWFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYV3/yUPSP+wXff+jbSt2sK7/5KHpH/AGC77/0baVu0AFFFFABRRRQAUUUUAFFFFABRRRQAVhN4ss18QHTDbXRUXAtGvQq+Qs5j8wRE7t2dpHO3bkgZzxW7XHXHhjVjrc3kGzbTX1NdVDPM6y+YsQAi2hCNpdQ2/OQCRtPWgDV1XxKdI1SztJ9H1CWK8uY7aK6iaAxl356GQPgAEn5eik80zS/GFnqusCwhtLuISef9nuJFTy5/IkEcm3DFhhiB8yrntmprrSbm+8UaXqFyYhaafBKyxKxLG4cBA3QDCp5gz1PmHgVheHfB19pXij+0p49PiI+0+fe2zN5+o+ZIGTzlKADYOnzN7bRxQBs+IP8AkOeF/wDsKP8A+kVzW7XM69ptivi7wzqK2VuL5tReNroRL5pT7Hc/KXxnHA4z2rpqACiiigAooooAKKKKACiiigAooooAKKKKACiisK88XafFctaacsurXq8G3sV8zb/vN91fxNVGMpbImU4x+Jm7Wdqmv6XooX+0ryOF3+5EMtI/0QZJ/AVmfYvEutc6heJolq3/AC72J8ycj0MpGF/4CPxrR0rw7pejFnsbVRO/37iQl5X+rnJNXywj8Tv6f5/8OZ805fCrev8Al/wxn/2/rGo8aJoEyIelxqb+Qo99gy5H4Cj+wNY1HnXNfmVD1ttNT7Og9t/LkfiK6Oij2lvhVg9lf4nf+vL9TL03w1o+kNvsNPhjlPWZhvkP1dst+talFFZyk5O7ZpGKirRVgooopFBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGFd/8AJQ9I/wCwXff+jbSt2sK7/wCSh6R/2C77/wBG2lbtABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBheIP+Q54X/wCwo/8A6RXNbtYXiD/kOeF/+wo//pFc1u0AFFFFABRRRQAUUUUAFFFFABRWbqviDS9GCjULtI5H+5CuWkf/AHUGSfyrN/tDxHrPGmWK6PbN/wAvOoDdMR6rCDx/wI/hWkacmrvRGcqkU7LV+RvXV3b2Nu097PHbwr96SVwqj8TWCfFM+pnZ4W0yW/B4+2T5htx7hiNz/wDAR+NS2vhCwW4W71aSbV7xeRNetvCH/ZT7q/gK3wMDA6U7047a/l/X3E2qS30/P/L8zm/+EYu9U+bxRqkl2h62VrmC3HsQDuf8T+FbtnZWun2y29jbxW8K9I4kCgfgKnoqZVJS0ZcacY6rcKKKKgsKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAwrv8A5KHpH/YLvv8A0baVu1hXf/JQ9I/7Bd9/6NtK3aACiiigAooooAKKKKACiiigAqhqWuaZo8lumq3sVn9oJWJ5zsRiMcbz8oJyMAnJ7Zwav1zfiZre71bT9Hv9RuLG0vILh5BFKkS3Gwxjyy5G4Eh2OEKnCtzxQB0YIZQVIIIyCO9eZXUtnD46mlneGPX01tGR3IEi6eLZSxz1EO3fn+Hfn+Kun8D6bo9ppclz4ciktbCV2gjtRMXhHkyyJ5qZz98AEnPICnrknp6AOK1/R9J1nx74fSPT7KS6Rzqk96IEMnlwgLEPMxnmR0I55EbVkeDLpo/GUkd7BplxqtzPfC7eKI/bbRFmOzzXJOY2XYFXCgDZjcOa9MooA5nXo77/AIS7wy7XNubH+0XCwC3YSh/sdzyZN+COvGz8a6asLxB/yHPC/wD2FH/9Irmt2gAooooAKKKKACiiigArO1/VhouiT3gTzJVASGIdZJGOEX8SRWjXNS/8T7xskX3rHRMSP6PcsPlH/AFOfqwrSnFN3eyM6kmlZbst+HPDtvo1kkksccupyruursqC8jnlvm64yeB6VtUUVMpOTuyoxUFZBRRRUlBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYV3/wAlD0j/ALBd9/6NtK3awrv/AJKHpH/YLvv/AEbaVu0AFFFFABRRRQAUUUUAFFFFABXGfEKDU7+K0sdPRXt9rXNwq20NxKypNCpCRyhgT5ckrfdPzKo74PZ1y3jDSNRv7vTrrR47oz2yyqXtr9bYhX2cHdG+4HaDjgAgHnjABe8Hfbx4UtBqxma4UyKGnhSF2jEjCMmNAAmU2nbjjoec1t1h+DryW+8LW81wZjMsk0UvnzrM4dJXRgXVVDYKkcDHHfrWJJrmqjxFJdC+YWUWtx6V/Z/lJtZGiUmTdt37977vvbdoxjPNAHb0VyviBtTi8VaHb6brd5Cb263TWYigaIW8S7pDkxl+TsTIbgyDGKy/DnjqPxB8RJ7O21mylsHtJDbWMckbSBo5ApdsfMGYbjt7KFOAc0AdD4g/5Dnhf/sKP/6RXNbtczr11MfF3hm1NhcLCNRdxeFo/KY/Y7n5QN+/P/Accda6agAooooAKKKKACiiigDP13VU0XRbi+ZfMeNcRRjrJIThVH1JAqLw3pT6RokUNw3mXcpM91J/flc5Y/nx9AKz7j/ifeNYrYfNZaLieX0e5YfIv/AVy31Iroppo7eFpZ5FjjUZZnOAPxrWbUIWfq/0/wAzKCc58y6aL9f8h9FZA8RQSjNnZ390p6PFbHafoWwKX+09Uk/1GhyAdjPcIn6DJrh+s0ns7+ib/JHb9WqrdW9Wl+bNaisnPiGXomnW4PqzyEfyo/s/V5P9drWwdxBaqv6tmn7dv4YN/cvzaD2KXxTS+9/kma1Izqi7nYKPUnFZX9gK/wDx86nqM3qDcFAfwUClXwzo6tuayWVvWZmk/wDQiaOeu9oL5v8AyTDkoreT+S/zaJ5tb0u3/wBdqNsp9PNGfyqv/wAJNpr/APHu09yf+mFu7/rjFXodPs7bH2e0gix/cjC/yqxRy4h/aS+Tf6r8gvQXRv5pfo/zMn+31P3NM1Nv+3Uj+eKP7bnb/V6JqJ/3lRf5tWtRR7Oq95/gv+CHtKXSH4v/AIBk/wBqam33NBn/AOB3Ea/1o+360fu6LGP9+8H9FNa1FHsZ/wDPx/8Akv8A8iHtYf8APtf+Tf5mT5+vv92xsY/9+5Y/yWj/AIqI/wDQMT/v43+Fa1FHsH1m/wAP0Qe2XSC/H9WZPk+IH+9eWEf+5buf5tR9j1wcjVrYn0Nnx/6HWtRR9Xj1b/8AAn/mH1iXZfcv8jJ8vxCvS506T/egcfyajd4iX/lnpj/8DkX+hrWoo9h2k/v/AMw9v3ivu/yMn7Vrqfe0y0k/3Lsj+aUf2jrA+9oef9y7Q/zxWtRR7Gf/AD8f/kv+Qe2h/wA+1+P+Zk/2vfL/AK3Qrwf7kkbf+zUf26w+/pGpr/2wDfyY1rUUeyqrao/uX+SD2lN7wX3v/NmT/wAJDAP9ZZajH/vWb/0FA8T6RnEl35J9Jo2T+YFa1BAIwRkeho5a62kvu/4Ic1DrF/f/AMAqQ6tp1x/qL+2k9lmU/wBatAhhlSCPUVVm0rT7j/X2NtJ7tEp/pVQ+GNIzlLMRH1hkZP8A0Eii+IXRP5tfowtQfVr5J/qjWorJ/wCEfVP+PfUtSh9ALksB+DZo/s3VY/8AU645HYTWyP8AqMUe1qrem/k1+rQezpvaa+af6JmtRWTs8Qx/dl06cf7SPGT+RNH23W4+JNIil94bsfyYCj6wl8UWvk3+Vw9g38Mk/ml+djWorPtNTmnuFhn0u7tic/O4UoPxDGtCtYTjNXiZThKDszCu/wDkoekf9gu+/wDRtpW7WFd/8lD0j/sF33/o20rdqyAooooAKKKKACiiigAooooAK5TxhZ6fcalpsuveHIdb0yOKdXJ00XskEpMewhdrMFIVwSo6hc11dcR491W60XV9GvdIja41IrJbpAYVdGilmt4yxJkj2kSNCAcn7x4xllAOh8LrMnh2BZ7FNPAeXybVIlj8qHzG8pSq8KfL2ZHY5pJvC+kT6w2qSW8n2tjuLJcSKu/Z5fmbAwXfsO0PjcBwDVPwE8kng+CS4SSO5e4umuUkVQUmNxIZFwrMAA+4AbjgAcnrXR0AVBploNSj1Dyy11Fbm2SVnZisZIYjk9yqknqcDmnHT7Y6oNRMX+lrCYBJuP3CwYjGcdQOetWaKAMLxB/yHPC//YUf/wBIrmt2sLxB/wAhzwv/ANhR/wD0iua3aACiiigAooooAKoa3qsei6Lc38q7/KT5EHWRzwqj3JIFPv8AVLXTlX7Q5Mj8Rwxjc8h9Ao5Ncu/2nxb4kNhqEH2ex03bPJGj5bzW+4rEd8Zbjpkdzxn7Vc/LFczW6X6vp+fZM09k+Tmb5U9m/wBF1/Lu0WtDmbSdNFjbRnUdXmcz3rIcIsznLF36DHQDrgDitOHRWuZludcmF5KpykIGIYj7L3Pua0rW1gsrdYLSJIYl6KgwKlrN05VZc9d38un/AAfn8ki1UjSjyUVbz6/8D5fNsOlFFFdBgFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYV3/wAlD0j/ALBd9/6NtK3awrv/AJKHpH/YLvv/AEbaVu0AFFFFABRRRQAUUUUAFFFFABVW+0uw1OJo9Rs4LpGjaMiWMN8rEEjnsSqn/gI9KtUUAV7GwtNMsY7PTreO2tohhIol2qvOTx9ST9TWU3iyzXxAdMNtdFRcC0a9Cr5CzmPzBETu3Z2kc7duSBnPFbtcdceGNWOtzeQbNtNfU11UM8zrL5ixACLaEI2l1Db85AJG09aANXVfEp0jVLO0n0fUJYry5jtorqJoDGXfnoZA+AASfl6KTzUlp4iW41waVc6be2M0kcktu9yI9twkbKrFdjsRjenDBTg9ODhLrSbm+8UaXqFyYhaafBKyxKxLG4cBA3QDCp5gz1PmHgVR0TTNdTxTfap4gt9PcyhorWW3vHcwQbgVjEZiUAtgM7biSQOwUAAteIP+Q54X/wCwo/8A6RXNbtczr2m2K+LvDOorZW4vm1F42uhEvmlPsdz8pfGccDjPaumoAKKKyrnWw07Wukwm+ul4bacRxf7z9B9BzWdSpCmryf8AXkaQpyqO0UaU08VtC0txIsUajLO5wBWR/aF9q3y6PH9ntj1vZ0+8P9hD1+p4qSHRTPMtzrUwvZ1OUjxiGI/7K9z7nmtasbVavxe6vx+/p8tfNGt6dL4fef4fd1+enkYd0lj4W0i71SRXnmRMtLKd0kzdFXPucAAcVL4Y0uXS9FX7aQ19dObm7f1lfkj6DhR7CqN//wAT3xjbacPms9J23d16NMf9Un4DL/8AfNdLXaoRo01CKtfX/L/M43OVao5yd7af5/5BRRRUFhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBhXf8AyUPSP+wXff8Ao20rdrCu/wDkoekf9gu+/wDRtpW7QAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYXiD/AJDnhf8A7Cj/APpFc1u1heIP+Q54X/7Cj/8ApFc1u0AUtQ086j5cb3M0VuM+ZFEdvm+gLdQOvA61YtraCzgWG1iSKJeioMAVLRUKnFSc7ast1JOKhfQKp6vqcOjaPc6hc8xwIW2jqx7KPcnA/Grlc1qX/E98XWulL81ppm28vPRpD/qkP6uR7Ct6cU5a7IwqScY6bvYueF9Mm07Rw99zf3jm5u2/6aNzj6KMKPpWzRRUyk5O7KjFRiooKKKKkoKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAMK7/5KHpH/YLvv/RtpW7WFd/8lD0j/sF33/o20rdoAKKKKACiiigAooooAKKKKACiiigArzG6ls4fHU0s7wx6+mtoyO5AkXTxbKWOeoh278/w78/xV6dRQBxWv6PpOs+PfD6R6fZSXSOdUnvRAhk8uEBYh5mM8yOhHPIjasnwj9m/4S2y+y+X/bP/ABMP7c2/6z/XDZ5vfrjy8/wZ2/LmvS6KAOZ16O+/4S7wy7XNubH+0XCwC3YSh/sdzyZN+COvGz8a6asLxB/yHPC//YUf/wBIrmt2gAooooAqarqUOkaTc6hdHEVvGXIHVvQD3JwPxqh4W02ax0gz34/4mF9Ibq7Po7fw/RRhfwqpqv8AxPPFdpo6/NaWG29vfRmz+6jP4gsR6KK6WtX7sLdXr/l/n9xjH35uXRaf5/5feFFFFZGwUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGFd/8AJQ9I/wCwXff+jbSt2sK7/wCSh6R/2C77/wBG2lbtABRRRQAUUUUAFFFFABRRRQAUUUUAFFFcRJrmqjxFJdC+YWUWtx6V/Z/lJtZGiUmTdt37977vvbdoxjPNAHb0VxF/4luk8b3NvNNqNppmnzWsDvawQtE7zYx5rOC+0l1UeWPl5LHBGIPDPiDWrzxckWoXE7WN39t8kyRw/Z5fLmAj+zlB5nCZ3eb16rkc0AdD4g/5Dnhf/sKP/wCkVzW7XM69dTHxd4ZtTYXCwjUXcXhaPymP2O5+UDfvz/wHHHWumoAKq6nqEGlaXcX9222G3jMje+Ow9z0q1XNaz/xPPE9noi/Na2m29vvQ4P7qM/VhuI9Fq6cVKWuxnUk4x036Frwrp89ppTXeoLjUNQkN1df7LN0T6KuF/A1t0UUpScpXZUYqMVFBRRRUlBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYV3/AMlD0j/sF33/AKNtK3awrv8A5KHpH/YLvv8A0baVu0AFFFFABRRRQAUUUUAFFFFABRRRQAVkTeF9In1htUkt5PtbHcWS4kVd+zy/M2Bgu/YdofG4DgGteigDJuvDGk3uoxX1zbyPPF5eP9IkCuYzujLoG2yFTyCwJB5FJYeF9I0zUmvrK2dJz5m0NPI6Rb23P5aMxWPcRk7QM1r0UAYXiD/kOeF/+wo//pFc1u1heIP+Q54X/wCwo/8A6RXNbtAFbUb+DTNNuL67bbDbxmRz7AfzrL8KWE8GmSX+oLjUNSk+1XAP8GR8qfRVwPzp+vaZc6xdadaFV/s1ZvPvCW5fZgomO4LYJ/3a2q1uowst3/X9fIys5VLvZf1/XzCiiisjUKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAwrv/koekf8AYLvv/RtpW7WFd/8AJQ9I/wCwXff+jbSt2gAooooAKKKKACiiigAooooAKKKKACiiigAooooA5rxdf2emah4ZutSu4LS3TVW3TXEgjRc2dyBljwOTirH/AAnPhP8A6GjRf/BhF/8AFVu0UAYX/Cc+Eh18UaLz0/4mEX/xVH/Cc+E/+ho0X/wYRf8AxVS6vJpba1osd9qdva3cN009rbyTKr3BMUkRCqTk8Sk8egrYoAwv+E58J/8AQ0aL/wCDCL/4qj/hOfCf/Q0aL/4MIv8A4qt2igDC/wCE58J/9DRov/gwi/8AiqP+E58J/wDQ0aL/AODCL/4qt2igDC/4Tnwn/wBDRov/AIMIv/iqP+E58J/9DRov/gwi/wDiq3aZLNHbwvNPIsUUalnd2AVVAySSegoAxf8AhOfCR6eKNF46/wDEwi/+Ko/4Tnwn/wBDRov/AIMIv/iqsaGthML7VNKv4r+21S5FwJoJFePKxRwkKykgj91+ea1aAML/AITnwn/0NGi/+DCL/wCKo/4Tnwn/ANDRov8A4MIv/iq3aKAML/hOfCf/AENGi/8Agwi/+Ko/4Tnwn/0NGi/+DCL/AOKrdooAwv8AhOfCf/Q0aL/4MIv/AIqg+OfCQ6+KNF/8GEX/AMVW7WP4nl0uPS4hreqW+mQC7t5VluJljVnilWULliBz5ePpmgCL/hOfCf8A0NGi/wDgwi/+Ko/4Tnwn/wBDRov/AIMIv/iq245EljWSJ1dHAZWU5DA9CDTqAML/AITnwn/0NGi/+DCL/wCKo/4Tnwn/ANDRov8A4MIv/iq3aKAML/hOfCf/AENGi/8Agwi/+Ko/4Tnwn/0NGi/+DCL/AOKrdooAwv8AhOfCf/Q0aL/4MIv/AIqj/hOfCWcf8JRouf8AsIRf/FVu1iwXOkJ40vFTVrVtUmtYYHsBOnmosZkcNszu5E3p0AoAZ/wnPhP/AKGjRf8AwYRf/FUf8Jz4T/6GjRf/AAYRf/FVu0UAYX/Cc+E/+ho0X/wYRf8AxVH/AAnPhP8A6GjRf/BhF/8AFVu0UAYX/Cc+E/8AoaNF/wDBhF/8VR/wnPhP/oaNF/8ABhF/8VW7RQBhf8Jz4T/6GjRf/BhF/wDFUDxz4SIyPFGikf8AYQi/+KrdrL8NR2UHhbTbbSr2O/s7a2S3iuYnV1lEahM5Xj+Ht3oArf8ACc+E/wDoaNF/8GEX/wAVR/wnPhP/AKGjRf8AwYRf/FVu0UAYX/Cc+E/+ho0X/wAGEX/xVH/Cc+E/+ho0X/wYRf8AxVbtFAGF/wAJz4T/AOho0X/wYRf/ABVH/Cc+E/8AoaNF/wDBhF/8VW7RQBhf8Jz4SHXxRovPT/iYRf8AxVH/AAnPhP8A6GjRf/BhF/8AFUviG80WzvdIn1vWLPTWtrl7iBbm4SLzj5LxEDcRnAmzx7etblAHJwa7pGs/ETTBo+q2V+YtKvfMFrcJLszLa4ztJxnB/KusoooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDz/wAV2V3JrmrwR2N1PNqtvYx2NxFbs6RNHM7NucDEewsJMsRnPGSMV6BRRQAUUUUAFFFFABUdw6R20rzKzxqhLqsZckY5AUAlvoASakooA5b4fXHmeG5Y2tby2dL+8cpd2ctuSslzLIhAkVcgqynjpnB5rqaKKACiiigAooooAKwfF2sz6NpUcljp895eTyiGExWklwsBIOZJBGpbaAD06nA4zkb1FAGX4asrfTfC+m2Vk07QW9skSNcRNFIwAxlkYAqfbAx6VqUUjuscbPIwRFBLMxwAB3JoAWisc+L/AA0LdJz4h0oQyFgkhvY9rFdobB3YON659Nw9RVxtX01NTj019QtVvpV3x2pnUSuvPITOSODzjtQBcorDvvGWg2McjnUrWfybmO2uBBPGxt2d9gMnzfIAc5z6GtWzvbXUbOO70+5huraQZSaCQOj844YcHkUAT1x2tmTUPG2irarqc32G83zWz2TJaqPKkBnExQZYBwAA5BzjbnkdjRQAUUUUAFFFFABRRRQBj+KbPU9Q0GSz0YW5kmdUmWedoQ0OfnUOqMVJXIzjjJPBxVD4eW1/Z+DorfVLCOwlS6udsEbsQEM7kfeRcDnjjkYPfA6eigAooooAKKKKACiiigDkPG6PDcWupafPq0OrW1rcR2a2Onm5jmLmM7Jf3bBQWjTksnGfmHUdXbNM1rE1yixzFAZEU5CtjkA/WpKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKy/E2jtr/hfUNKjm8h7uBo1k5wCfXBBx2OCDjNalFAHlx8Ka1aeMrS7s9FsTPNpl9FO91ql1eREsbVV3SyRkq2FbCY5VWGfTTsvAmpWDQ2CXFrNYi7sryS9d2FyrW8MUewLtIIbyR824YEjjB6nvqKAOJtPCmrL4X0nQbuLTfI0i4smiuEmdjcJBIrFmQxgIxCg4DMCxPI610Hh/SptJhv0naMi4v57mMRk4CyPuAOQOeTmtaigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigD/9k=)

![A close up of text on a white background

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDcRXhpZgAATU0AKgAAAAgABAE7AAIAAAAGAAAISodpAAQAAAABAAAIUJydAAEAAAAMAAAQyOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEhhcnNoAAAFkAMAAgAAABQAABCekAQAAgAAABQAABCykpEAAgAAAAM5NAAAkpIAAgAAAAM5NAAA6hwABwAACAwAAAiSAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAxOToxMDowNyAxMjozNzozMwAyMDE5OjEwOjA3IDEyOjM3OjMzAAAASABhAHIAcwBoAAAA/+ELGGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTAtMDdUMTI6Mzc6MzMuOTM5PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkhhcnNoPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIASoB1wMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APpGiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKoanrukaKYhrGq2WnmbcYhdXCReZtxnG4jOMjOPUVR/4Tnwn/ANDRov8A4MIv/iqAN2isL/hOfCf/AENGi/8Agwi/+Ko/4Tnwn/0NGi/+DCL/AOKoA3aKwv8AhOfCf/Q0aL/4MIv/AIqj/hOfCf8A0NGi/wDgwi/+KoA3aKwv+E58J/8AQ0aL/wCDCL/4qj/hOfCf/Q0aL/4MIv8A4qgDdorC/wCE58J/9DRov/gwi/8AiqP+E58J/wDQ0aL/AODCL/4qgDdorC/4Tnwn/wBDRov/AIMIv/iqP+E58J/9DRov/gwi/wDiqAN2isL/AITnwn/0NGi/+DCL/wCKo/4Tnwn/ANDRov8A4MIv/iqAN2isL/hOfCf/AENGi/8Agwi/+Ko/4Tnwn/0NGi/+DCL/AOKoA3a5jxHq+tweKNG0XQWsIjqFvdSyz3kLy+X5XlbcKrrnPmEEZHY54wbX/Cc+E/8AoaNF/wDBhF/8VVWbxN4Hn1S11KXxForXdpHJFDJ/aUY2LJt3jG7BzsXqO3HegDC0nx7rTWmj6nrVtYiy1S2uW+z2qOZYZIIy5O8thgwR8DaCuRy3NNn8f61omlRahrttYXMd9o0+qWsVkHQwtGI28p2ZmDgiVf3gC4wfl5FaltqXw7tItOjt9c0VU0wubQHVEPl7wVbq/wA2Qx6561VsD8MdNWdbbV9EKT27WrRzassyJA3WJFeQhEP9xQBwOOKAMfUtd1nwv4w1XU9dWyvrq18PRvELGKSNXL3JUKVJduGP3hnI6KDxSaz4i1TVvDeoWeqW0jJDd6ZLFejSbmwjctexho9k/JZdoOQSCGHAxWvZr8MLFrlo9Z0iZrq2+yTm61n7QZIc5EZ8yRvlGeB27VJHP8Oo7OW1bxJYzwzPFIy3XiBpyDE4dMF5SVAYA4GB60AaHjj/AFnhr/sO2/8A6C9Zum+MtbuW0fULi3sX03Xbqa2tLaJWWeEqkrxl3LFW3CEggKu0sOTg1q33ibwPqRtje+ItFlNrOtzD/wATKMbZFzhuG56ng8Vm283w1tNSe+g1rR1mYyMoOrgxxGTPmNHGZNkZbJyVAJyfWgCvYeMPEU+iX7PFby63DDDL/ZQ0ueCW2DOFdsPIftCqNxBj2hiuBgkV03hXV31nSXmmvre8mimaKVobKW0MbAA7HhlZnRgCDgnoQe9c7CfhvBDcIniOyZrlEjeaTxE7zKqtuVUlaYugDc4UgVo6Rr/gTQ7V7fTfEekIsshlleXVllklcgAszu5ZjgAZJPAA7UAdZRWF/wAJz4T/AOho0X/wYRf/ABVH/Cc+E/8AoaNF/wDBhF/8VQBu0Vhf8Jz4T/6GjRf/AAYRf/FUf8Jz4T/6GjRf/BhF/wDFUAbtFYX/AAnPhP8A6GjRf/BhF/8AFUf8Jz4T/wCho0X/AMGEX/xVAG7RWF/wnPhP/oaNF/8ABhF/8VR/wnPhP/oaNF/8GEX/AMVQBu0Vhf8ACc+E/wDoaNF/8GEX/wAVR/wnPhP/AKGjRf8AwYRf/FUAbtFYX/Cc+E/+ho0X/wAGEX/xVH/Cc+E/+ho0X/wYRf8AxVAG7RWF/wAJz4T/AOho0X/wYRf/ABVH/Cc+E/8AoaNF/wDBhF/8VQBu0Vhf8Jz4T/6GjRf/AAYRf/FUf8Jz4T/6GjRf/BhF/wDFUAbtFUNM13SNaMo0fVbLUDDt80Wtwkvl5zjO0nGcHGfQ1foAKKKKACiiigAooooAKKKKACiiigAooooAKKwm8WWa+IDphtroqLgWjXoVfIWcx+YIid27O0jnbtyQM54qxLryjXf7LtLC7vJIwhuZoPLEdsHzt3lnUk4BOFDEDBI5GQDVormR480sPdNcw3VtawQ3E0d5Ki+VcLbttlKYYtwf7wG4crkc1paLrsesNcxNZXVhdWrKJrW7CB1DLuVvkZlIIz0PUEHBFAFa7/5KHpH/AGC77/0baVrCWZ9xiiQqGK5ZyCcHB4waybv/AJKHpH/YLvv/AEbaVYu7zUbS3Q6Zpf8AaBaWTePtCxbPmOOvWmld2E2krsv7rn/nlF/39P8A8TRuuf8AnlF/39P/AMTXPPrPiOKbzG8L4V8KR/aMfXsenFTf2z4k/wChV/8AKjH/AIVp7KXl96/zMvbR7P7n/kbe65/55Rf9/T/8TRuuf+eUX/f0/wDxNJZS3E9nHJeW32Wdh88PmB9nP94cGuCTW/FS+Mr3TrXUNOa3uNaktIvtNlI7WyLZJOMYlG4Z4xxyScjoM3o7Gqd1c77dc/8APKL/AL+n/wCJo3XP/PKL/v6f/ia861P4gain9tRrDHLZHS9SubCfyzAWa2IUjKzmQjJI3bYjkZHXjWXxTrI8QTK62P8AZkWsRaZ5YifzmD28cgffvxwz4xt5HcGkM7BJX84RyoqkqWBVsg4xnsPUVLULf8f0X/XN/wCa1NQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYVp/wAlD1f/ALBdj/6Nu63awrT/AJKHq/8A2C7H/wBG3dbtABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHHXHhjVjrc3kGzbTX1NdVDPM6y+YsQAi2hCNpdQ2/OQCRtPWoNU8E3d34rfVbaKxV7qe0uHvHlfz7Qwld6xAJhg6qFPzL1OQwwK7iigDz+bwFqd9YyaVd3FpFY29tew2U8Ts8rmc/IzoVAXYpIIDNuPPy9K6PQNN1KHUtR1XW1tYru9EUYhtJWkREjBx8zKpJLO56cDA5xmt2igDmX02xtPidYXVpZW8FxdaZemeaKJVeYiW1wWIGWxk9fWugtf9S3/AF0f/wBDNZF3/wAlD0j/ALBd9/6NtKuyf2h5A/s37NnzZN/2jd/eOMYqZS5Ve1yox5na5ekQSRsjdGGDUdu7FTHJ/rIzg+/oaz/+Ki/6hf8A5EqGYeIIm+0H+zflGGx5nI/+tWSr9OV/cauh15l95uVS/sXSv7X/ALV/syz/ALRxj7Z9nXzsY2/fxnpx16VYtjObZDd+X52Pm8rO38M815Pqeu6po/i7WbS38RrarPqzGRb26ggS2i+ywsjBmgkKhmyoLAqduPvHJ2TurmLVnY9KHhvQxPczDRdP827V1uJPsqbpg/3w5x8wbvnr3q1/ZtjuLfY7fc0wnJ8peZAAofp94AAZ64ArgYfEmqLrui22o+ILeb7bbwx+To80Dl5GDZmKSR72jPykMhAGDlcAmtr4Yzeb8PNKEmqyalPHbxpcGV42a3kEa7oTsUYKns2W55JpiOnb/j+i/wCub/zWpqhb/j+i/wCub/zWpqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDCtP+Sh6v/2C7H/0bd1u1hWn/JQ9X/7Bdj/6Nu63aACiiigAooooAKKKKACiiigAooooAKKKKAMJvFlmviA6Yba6Ki4Fo16FXyFnMfmCInduztI527ckDOeKlHiDzPEs2j2+l3s5t1jae6QxCKIOCVzukDn7p6Kaxrjwxqx1ubyDZtpr6muqhnmdZfMWIARbQhG0uobfnIBI2nrUur+G7vU/EVpfQ6bpVpLDPBK+qpMxu9iMGaEDyxlWG5OXxtYnbmgCxceMkstTuLK+0XUrYwWU995rGBkeKIgEjbKSCdw2hgM89MGtXR9Tk1axF1Jpt3p4bBRLpoizqQCGHlu4xz3IPtWeuiXB1bXtRuo7W6e9gS1tIJGOzyEQna52nG6SSTOAfl29cYqPwnoFxos2oSyWljpsF06GLTtOcvBDtXBYEonzNxkBQPlHU5NAE13/AMlD0j/sF33/AKNtK1YzJCGTyHcb2YMpXBySe5HrWC+nw2vxOsLiN7hnuNMvS4luZJEGJbX7qMxVOv8ACBmumoAh86T/AJ9pfzT/AOKoMzkEG1lIPXlP/iqmooAyrfVY49SfSmjl+0pF5yIcDMecZBzg4PHHtV/zpP8An2l/NP8A4qsfxTYztbwavpqbtQ0tjNGg/wCW0eP3kf8AwJenuBWtp99BqenwXtm++CdA6N7H+taSiuVSX9MzjJ8zi/6Q7zpP+faX80/+Ko86T/n2l/NP/iqmorM0IF3yXKyNG0aqhX5iMnJHoT6VPRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYVp/yUPV/+wXY/+jbut2sK0/5KHq//AGC7H/0bd1u0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFZbeJNKXXRo5uW+2lhHt8l9gcpvCeZjYH2fNszuxzjFK3iHT111tHBuXvFCFxHZzOkYbO3dIFKLnB6kUAadFZFn4o0q+a88iWcR2LTLPPLaTRQqYnKSYlZQjbWVhwT0PpU2j67p+uwyyabLI3lMFkSaCSF1yAwJR1DYIIIOMHtQBTu/wDkoekf9gu+/wDRtpW7WFd/8lD0j/sF33/o20rdoAKKzb2zvluTeaZdYkIAe2nOYnx6d1PuKLLWYrif7LdRtZ3uOYJf4vdT0YfSsPbJS5Jq3bs/n+m5t7FuPNDXv3X9d9jSrmNM/wCKe8TzaO3y2Oolrmw9Ek6yxf8AswHua6esrxHpLavpJjtn8q9gcT2kv/POVeVP0PQ+xNddNq/K9mclSLtzR3X9WNWis7QdWXW9HivAnlS8pPCesUinDKfoRWjUSTi7M0jJSV0FFFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYVp/yUPV/+wXY/+jbut2sK0/5KHq//AGC7H/0bd1u0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAcNc6Nq667PBHpskto+spqwvUljAKLCo8raWDeYXTaONu05LDpVnVNMvZfFUF9o2m6laXck9sbm+N6otXgUgyI8IkO5thdQdnDEHdgV2FFAHnd74X1G/Gt6fpFjfaTY6hY38c4vr1ZoZ7idtyvGgkfYNzSMeE4bGCenReHrfUJ9e1TWdS06TTPtUNvbx2sssbv8AuvMJclGZcEykDnOFyQM4roqKAOZfT4bX4nWFxG9wz3GmXpcS3MkiDEtr91GYqnX+EDNdNWFd/wDJQ9I/7Bd9/wCjbSt2gAqve2FtqMHk3kKyLnIz1U+oPUH6VYoqZRUlyyV0OMnF3i7Mxf8AiZ6N/wA9NUsh/wB/4x/7OP1+taVlf2uowebZzLIucHHBU+hHUH61YrOvdGiuZ/tVrI1negcTw/xezDow+tc/JUpfw9V2f6P9H96Ojnp1fj0fdfqv1X3MyZ/+Kc8XJcj5dO1pxHN6RXIHyt7bwNp9wK6eua1ST7Tpk+leKIhDFcLsW9h5j3fwt/sMDg88ZHWrXhbVZdQ017e/K/2jYv5F1joxA+WQezLhh9TXXGrGvDmW63XXy/y+45JUpUJ8r2eqfTz/AM/vNuiiikUFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGFaf8lD1f/sF2P8A6Nu63awrT/koer/9gux/9G3dbtABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAFQ6tpw1UaYb+1GoMnmC0My+aV/vbM5x74qOXXdIg1ZNLm1Wyj1CQApaPcIJWz0wmcnOD2rjLm2uYvEtxbtYXjTnXo9TW7S1do/s62yqT5gBG7CtFszuOeBg5q7qJuLTxkl3oUmqvd3s1rHdWr6e32RoAcO5mMY2sqMxxv+8ACvNAHSweINGupLpLbVrGZ7ME3Sx3KMYAOpcA/LjB6+lSabrGmazA0+j6jaX8SNtaS1nWVVOM4JUnnBrjBJD4k17VZL6x1OwtrSwutPtoxpc6NLG5XzZQxj2sSUXYgJJAJwS2F2fBs18y3tvLJfXGm25jSyudRs/s07/L86lNiZVTjDbRnJ64zQBZu/wDkoekf9gu+/wDRtpW7XMvazQfE6wmkv7i4SbTL0pBKsYSDEtrwpVAxz/tFuldNQAUUUUAFFFFACOiyIUkUMrDBVhkEVx2tWP8AwiFwNe0aPEBKxXkDMdgjJ4b2AJ/DJ7ZB7Ko7iCK6tpLe4QSRSoUdG6MpGCKiVOE2nL71o/6/DuUqk4pqP47f1+PYpWGs297J5Dhra7Ay1vLw2PUdmHuK0a5PRLWKRbjwzrI82fTMNazE4d7c/ccN1yuNpx3A9a0vM1PRuJw+p2Q/5aIP38Y9x/GPcc1nN1KErVNV3X6r9V+BpBU68eano+z/AEf6P8TaorJ/4SfSh96aVf8AetpB/wCy0f8ACUaN3vlX/eRh/MUvrWH/AJ196K+q1/5H9zNaistfE2it01K3/F8VIuvaQ3TVLP8AGdR/WmsRRe0196JeHrLeD+5mhRVRdW05vu6han6TL/jUi31o33bqE/SQVaqQezRDpzW6ZPRTFmjb7sin6MKfV3TJs0FFBIHU4ppljHV1H40XQWY6iojcwDrNGPq4ppvrQdbqEfWQUueK6j5Zdieiqx1KxHW9tx9ZV/xph1fTR11C1H1mX/Gp9rTX2l95Xs6j+yy5RVE63pQ66nZj/t4X/GopPEejR/e1K2P+7IG/lUvEUVvNfeilQqvaL+5mnRWT/wAJNpjf6l5pj6RW0jf+y0f2/u/1WlanJ/27bf8A0Iip+tUOkk/TX8ivq1brFr10/M1qKyf7U1J/9ToU+PWWeNP6mj7Tr0n3NNs4f+ut0W/ktH1iHRP/AMBf+QfV59Wv/Al/ma1FZOzxC/WbTYR/sxu5/Uij7BrL/f1pU9orRR/Mmj20ukH+H6tB7GPWa/H9EzWorJ/sW4f/AF+t6g3+4UT+S0f8I1pz/wDH19ouj6z3Dv8ApnFHtKz2h97/AMrhyUlvP7l/nYuz6lY22ftN5bxY/vygf1qkfE2lE4huGuG9IInk/kKswaNpltjyLC2Q+oiGfzq6AAMAYHoKLYh9Uvk3+qC9BdG/ml+jMn+25pOLXR9QkPYyIsQ/8eI/lR9q12b/AFWm2tv7z3Jb9FX+ta1FHsqj3qP5W/yv+Ie1gtoL53/zt+Bk/Zddm/1upWtv7QWxb9Wb+lH9hzSc3WsahIe4SRYh/wCOgfzrWoo+rU3vd+rf+dg+sT6WXol/lcp2Ol22nFzbiQtJje0krOTj6k+tXKKK2jCMFyxVkYylKb5pO7MK0/5KHq//AGC7H/0bd1u1hWn/ACUPV/8AsF2P/o27rdqiQooooAKKKKACiiigAooooAKKKKAKWoaxp+lKx1C6SDbBJcHdn/Vx7Q7cehdR/wACFSadqNrqthHeWEvmwSZCsVKkEEqQVIBBBBBBAIIINcp47MA1DS/N0qa+n2S/ZGSSVUM/mQbIpNnBRjhzu4xCTjrWz4RLHw8PNt1t5ftd15qqzFXk+0Sb3XdztZsuB2DCgDboqodW04aqNMN/ajUGTzBaGZfNK/3tmc498VHLrukQasmlzarZR6hIAUtHuEErZ6YTOTnB7UAX6Kpw6xptze3Nnb6jaS3VoM3ECTqzw/76g5X8adp2qafq9r9p0m+tr6DcV822mWRcjqMqSM0AZl3/AMlD0j/sF33/AKNtK3awrv8A5KHpH/YLvv8A0baVu0AFFFFABRRRQAUUUUAc94qtZoFt9e09C95phLPGvWeA/wCsT645HuK27S6hvrOG6tXEkMyB43HcEZFTVzOif8U/4gn8PyfLaXG6600noBnMkQ/3SdwHo3tWy9+Fuq/L+v1MX7k79H+f/B/yOmooorE2GtGjfeRT9RUbWdq33raE/WMVNRUuKe6KUmtmVG0rT2+9YWx+sK/4VG2haS3XTLP/AL8L/hV+iodGm94r7i1WqLaT+8zG8N6M3XTLb8IwKZ/wi2i/9A+IfQkf1rWoqPqtB/YX3Ir6zXX2397MkeF9EH/MNh/EE04eGtFH/MNt/wAUrUoo+q0P5F9yD6zX/nf3szh4e0cf8wy1/GEU4aFpA6aXZ/8AgOv+FX6Kr6vRX2F9yF7er/M/vZSGjaYOmm2g+kC/4U8aXp46WNsPpCv+FWqKr2VNfZX3E+1qP7T+8rjT7MdLSAfSIf4VLHBFF/qokT/dUCn0VShFbIlzk92FFFFUSFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGFaf8AJQ9X/wCwXY/+jbut2sK0/wCSh6v/ANgux/8ARt3W7QAUUUUAFFFFABRRRQAUUUUAFFFFAHn/AI917RodWtLa91rTWS1Egu9KfXUsZS7hDG5+ZSQBu+ViAd4bnAro/BUpm8I2cn9oQagrGUpPBd/akCeY21PO/wCWhRcIWPJKnNbtFAHntzbXMXiW4t2sLxpzr0eprdpau0f2dbZVJ8wAjdhWi2Z3HPAwc1d1E3Fp4yS70KTVXu72a1jurV9Pb7I0AOHczGMbWVGY43/eABXmu1ooA8mutF1S/wBHl0qz067S/tLHVo7mV4WjS4aZ8qqSEBX8w4fgnGPmweK7LwwTd69rWqQWdzZ2NyLeOJLm2e3Z3RCHbY4BAwUXJHOzjgCunooA5l475fidYNd3NvLbtpl75EcVuyPGPNtc7mLkN26Kv4101YV3/wAlD0j/ALBd9/6NtK3aACiiigAooooAKKKKACsjxJpMuqaarWLLHqFpILizkPQSDsfZhlT9a16KqMnF3RMoqSszE0zxRa3l0LDUIpNM1PHNpdcF/dG6OPofwrbqpqWlWOsWhttStY7iI8gOPun1B6g+4rE+x694e50yVtasF/5dLl8XEY/2JDw30bn3rTlhP4dH/XX/AD+8z5pw+LVd/wDgf5fcdNRWXpHiHT9ZLx20jR3UX+ttJ18uaL6qefxHFalZSi4uzNIyUldBRRRSKCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDCtP8Akoer/wDYLsf/AEbd1u1hWn/JQ9X/AOwXY/8Ao27rdoAKKKKACiiigAooooAKKKKACiiigAooooAKK8xupbOHx1NLO8MevpraMjuQJF08WyljnqIdu/P8O/P8VTeJ5tRm+Inh66m0i8msYNSjjsJ4ZoDFIHtpTJJgyBt3OOV4WNsE78EA9Iorzmfw5ot9r/i25traw0m3ttLk0yS8jgSMCWZPNnkcjG4Kph5J/v8APWtb4ezWgtdQsdMi0l7S1mQJfaNbiG3uiUBPAZhvXABIY9unQAGnd/8AJQ9I/wCwXff+jbSt2uZeO+X4nWDXdzby27aZe+RHFbsjxjzbXO5i5Dduir+NdNQAUUUUAFFFFABRRRQAUUUUAFFFFAGbq/h/T9aCNeRFZ4uYrmFiksR9Vccj6dKy/tOv+HeL2Ntd09f+XiBALmMf7SdH+q4PtXTUVpGo0rPVGcqab5loynpmr2Gs2v2jTLlLiPOG2nlD6MDyD7GrlYup+GLO/uvt1q8mnakBxeWp2sfZh0cexBqn/buqaD8nii1861HTU7JCUA9ZI+Sn1GR9KrkUvg+7r/wf60J9o4fH9/T/AIH9anTUVFa3Vve2yXFnNHPDIMrJGwZW/EVLWOxtuFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYVp/wAlD1f/ALBdj/6Nu63awrT/AJKHq/8A2C7H/wBG3dbtABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYV3/AMlD0j/sF33/AKNtK3awrv8A5KHpH/YLvv8A0baVu0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHPXXhVYLl73w3ctpN2x3Osa7oJj/ALcfT8Rg0xPFUmmMIPFlmdOfot3HmS2k+j4yn0bH1rpKRlDKVYBlIwQR1rX2l9Jq/wCZj7O2sHb8vuGW9zBdwLNazRzxMMrJGwZT9CKkrAuPB2nee1zpLz6PdMcmSwfYrH/aT7rfiKi+0eKNI/4+raDXLcf8tbXEM4HqUJ2t+BH0o5Iy+F/fp/wA55R+Nfdr/wAE6SisbTvFWk6jcfZluDbXne0u0MMoP+63X8M1s1Eoyi7SRpGUZK8WFFFFSUFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAYVp/yUPV/+wXY/+jbut2sK0/5KHq//AGC7H/0bd1u0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAefy3t3/AMJVLeC+uhcx+II9OSzE7eUbc26sV8rO0nDNLuxu464GKb4l8Q3cfxF0W1J1a0s7fUEg2w2dx5V4ZIJGLF1XYyqdgAzwQ5IwoI7SXRNKuNQN/PplnLeNEYTcPbqZDGRgpuIztwTx0qw9pbSCAPbxMLdg8OUB8pgCoK+hwSOOxNAHm93d6t4Vsteju3mfUriw1G/srxdUmuUWNJNy5gfCxlFljA2gg7SM9j0/hjdaa9rWlwXlzeWNsLeSJ7m5e4ZHdCXXe5JIwEbBPG/jgitiz0LSNOmnl0/SrK1kuf8AXvBbojS85+YgfN1PX1qXTtL0/SLX7NpNjbWMG4t5VtCsa5PU4UAZoAw3kvm+J1gt3bW8VuumXvkSRXDO8g821zuUoAvbozfhXTVhXf8AyUPSP+wXff8Ao20rdoAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAqajpVhq1v5Op2cN1H2EqA4+h7fhWN/wAI3qGl/N4b1mWKMdLO+zcQ/QEneo+hNdJRVxqSirLYzlTjJ3e5zf8Awk95pny+JdHntUHW7tM3EH1OBuX8R+NbVhqdjqluJ9Nu4bqL+9E4bH1x0q1WLf8AhPSb64N0kLWV52u7JzDJ+JXr+INVenLdW/r+upNqkdnf1/r9Daorm/K8U6R/qJrfXrcfwTYguAPQMPkb8QKfF400xJFi1dbjR5icbL+IxqT7Pyh/Oj2Un8Ovp/Vx+1ivi09f89joaKZDPFcRLLbypLG3KujBgfxFPrI1CiiigAooooAKKKKACiiigAooooAKKKKACiiigDCtP+Sh6v8A9gux/wDRt3W7WFaf8lD1f/sF2P8A6Nu63aACiiigAooooAKKKKACiiigAooooAKKKKACivP5b27/AOEqlvBfXQuY/EEenJZidvKNubdWK+VnaThml3Y3cdcDFN8S+IbuP4i6Lak6taWdvqCQbYbO48q8MkEjFi6rsZVOwAZ4IckYUEAHoVFcH9ku77U9ch8MajfC3XT7m1lubm+lkjN8xGwxFidnl4cMUwAXA5KkDW8HTNCNQ0m6hmjvbCRPPL6lLfK+9AQVkl+YcDlcDHXvkgFm7/5KHpH/AGC77/0baVu1zL3U03xOsIZLC4t0h0y9CTytGUnzLa8qFcsMf7QXrXTUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU2SNJo2jmRZEYYKsMg/hTqKAOfm8F6QZWm05ZtJnbrLp8phz9VHyn8RTPsvivTf+Pa+s9YiH/LO7j8iXHoHTKn8Vro6K19rL7WvqZexj9nT0Oc/4S9bLjxBpV9pWOsrR+dD/AN/Ez+oFbNhqljqkPm6deQXSd2hkDY+uOlWqxr/wnomoTefLYpFcdRcW5MMgPruXB/Oi9N7q34/194WqR2d/w/r7jZornP7F1/TudI143UY6QapF5n/kRcN+eaP+Ei1Sw41zw/coo63GnsLmP6lRhwPwNHs7/C7/ANeYva2+JNf15HR0Vmab4k0fV22afqEEsneIttkH1Q4YflWnWcouLs0aRkpK6dwooopFBRRRQAUUUUAFFFFAGFaf8lD1f/sF2P8A6Nu63awrT/koer/9gux/9G3dbtABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAFGXRNKuNQN/PplnLeNEYTcPbqZDGRgpuIztwTx0qw9pbSCAPbxMLdg8OUB8pgCoK+hwSOOxNTUUAZNv4U8PWjzva6DpkLXKNHO0dnGplVuWVsD5ge4PWrmn6XYaRa/ZtKsbaxg3FvKtoVjXJ6nCgDNWqKAMK7/wCSh6R/2C77/wBG2lbtYV3/AMlD0j/sF33/AKNtK3aACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAKGpaFpWsLjU9Pt7k9meMbh9G6isz/hEFtedG1jVNOx92Nbjzox/wCTdXRUVoqk4qyZnKnCTu1qc75Xi+x/1dzpmrIO0sbW0h/Fdy/oKP+EovLTjWPDmpW3rJbBblB75Q7v8Ax2uiop+0T+KK/L/gfgL2bXwyf5/8H8TGsvF2gX8nl2+qW4l6eVK3lP8A98tg1sgggEHIPQ1WvdNsdRj2ahZ290vpNEH/AJ1jHwTpcJLaVLe6U3/TldMi/wDfByv6UWpvuvx/y/IL1V2f4f5/mdFRXO/2d4psv+PLW7W/XtHqFrtP/fcZH/oNH9u65Zf8hTw1NIo6y6dOswP/AAFtrfzo9nf4Wn/XnYPa2+JNf15XOiorN0nXbPWGlS2W4jlhx5sVxbvEyZzj7wHoelaVZuLi7M0jJSV0YVp/yUPV/wDsF2P/AKNu63awrT/koer/APYLsf8A0bd1u0hhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHESa5qo8RSXQvmFlFrcelf2f5SbWRolJk3bd+/e+7723aMYzzUGv+Oo7b4haVottrNlZxRXqW99BLJGJZ2khd1UBuQo/d/MOrSKAeCD083hfSJ9YbVJLeT7Wx3FkuJFXfs8vzNgYLv2HaHxuA4Bq7caba3TWZuIzI1lKJoCztlXCsmc55+VmHOetAHnk3i/XLDSRqzXouP7SsL24htZIUCWbxOojwVAYqA2H3E5IGNvSut8O3N9Hq2r6TqF9JqP2FoXiupY0RysiZKsEVVyCpOQBww64yZovCGhwz3kq2O77bHLFMkkrvHslO6RVRiVQOeWCgZPJzVvSdEsdEgli0+ORfOfzJXmneaSRsBQWdyWOAoAyeAAKAMh7qab4nWEMlhcW6Q6ZehJ5WjKT5lteVCuWGP9oL1rpqzNU8P2GsXVvc3n2pJ7ZHjiktb2a3YK5UsCY3XIJReuelVf8AhD9N/wCfnWv/AAeXv/x2gDdorlYvBkY166eW81g6ebaEQJ/bl5kShpfMP+tzypi/L61d/wCEP03/AJ+da/8AB5e//HaAN2isL/hD9N/5+da/8Hl7/wDHaP8AhD9N/wCfnWv/AAeXv/x2gDdorC/4Q/Tf+fnWv/B5e/8Ax2j/AIQ/Tf8An51r/wAHl7/8doA3aKwv+EP03/n51r/weXv/AMdo/wCEP03/AJ+da/8AB5e//HaAN2iuV0bwZHHoNgmsXmsSagttGLp11y8w0u0bzxLj72elXf8AhD9N/wCfnWv/AAeXv/x2gDdorC/4Q/Tf+fnWv/B5e/8Ax2j/AIQ/Tf8An51r/wAHl7/8doA3aKwv+EP03/n51r/weXv/AMdo/wCEP03/AJ+da/8AB5e//HaAN2isL/hD9N/5+da/8Hl7/wDHapX3gyNrzTTY3msLAtyTeA65efNF5MgAGZf+ehjPHp9aAOqorC/4Q/Tf+fnWv/B5e/8Ax2j/AIQ/Tf8An51r/wAHl7/8doA3aKwv+EP03/n51r/weXv/AMdo/wCEP03/AJ+da/8AB5e//HaAN2isL/hD9N/5+da/8Hl7/wDHaP8AhD9N/wCfnWv/AAeXv/x2gDdorC/4Q/Tf+fnWv/B5e/8Ax2qVj4MjW81I315rDQNcg2YGuXnyxeTGCDiX/noJDz6/SgDqqKwv+EP03/n51r/weXv/AMdo/wCEP03/AJ+da/8AB5e//HaAN2isL/hD9N/5+da/8Hl7/wDHaP8AhD9N/wCfnWv/AAeXv/x2gDdorC/4Q/Tf+fnWv/B5e/8Ax2j/AIQ/Tf8An51r/wAHl7/8doA3aK5XWfBkcmg36aPeaxHqDW0gtXbXLzCy7TsPMuPvY61d/wCEP03/AJ+da/8AB5e//HaAN2isL/hD9N/5+da/8Hl7/wDHaP8AhD9N/wCfnWv/AAeXv/x2gDdorC/4Q/Tf+fnWv/B5e/8Ax2j/AIQ/Tf8An51r/wAHl7/8doA3aKwv+EP03/n51r/weXv/AMdo/wCEP03/AJ+da/8AB5e//HaAN2iuVi8GRjXrp5bzWDp5toRAn9uXmRKGl8w/63PKmL8vrV3/AIQ/Tf8An51r/wAHl7/8doALT/koer/9gux/9G3dbtZml+H7DR7q4ubP7U89yiRyyXV7NcMVQsVAMjtgAu3THWtOgAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDz+W9u/+EqlvBfXQuY/EEenJZidvKNubdWK+VnaThml3Y3cdcDFegVUOk6cdVGqHT7U6gqeWLswr5oX+7vxnHtmrdABRRRQAUUUUAFU9WtFvtJuLeSWeFXTl7eZopBjnh1IYdOxq5UF7Y2mpWclpqNrDd20oxJDPGHRxnPKng8igDM8GTzXXgPQLi6leaaXTLZ5JZGLM7GJSSSeSSec1tVV07S9P0ezW00ixtrC2UkrDawrEgJ6naoAq1QAUUUUAFFFFABXJeKEMuv6fBpN3fDWZJYpBHFdyLDDbJIDLJJEDsIK7kG4EliAMYJHW1m3fhzQ9Q1FNQv9G0+5vUChbma1R5FCnK4YjIwSSPTNAGlRRRQAUUUUAFFRyXEEM0UUs0aSTMViRmAMhAJIUdzgE8dhUlAASADngdzXH+G0utP8barpk8swtvskU9tDJqEt7uXzJFMheX5o2OFGwZXjIJ5x17KGUqwBBGCCOtU9N0XS9FSRNH02zsFkILrawLEHI6E7QM0AXaKKKACiiigAooooA5fx5qlzYaItvaLfp9sLJLd2NrLM1tGFJYjy1JVjjaG4wW3fw1e8GXraj4G0S7keaSSWwhZ3nVld22DJO4ZOTk5PXrzmtllV1KuoZWGCCMgimwwxW0EcFvGkUUShI40UKqKBgAAdAB2oAfRRRQAUUUUAFFFFAHFyreaP4+N9qZmuLLU75ILEx6rNiA/ZgCrWvEZG5JG3ZJBYHHGR2lUU0TSo9WbVU0yzXUXGGvBboJmGMcvjPQAdavUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFc744Gqnw0RoryxyG4i894Y3kkWHePMKqjK5OOysGxnHOK6KigDxmOTWU1bU7SHVPEV/eR6bayaUEt7u3jjme4uQPNjkZm2/KoLTEgqvJ4Wtm5/4Sn7Vf/2T/a39tbtR8zz/ADPsflbZPsnl7v3W7Pkfd5+/u716StvAt09ysMYnkRY3lCjcyqWKqT1IBdiB23H1NSUAeevpNtq114eSwHiIWsd9K11JeSXkUyH7K4/1khDopJUZQhSeB1NdT4Ra+bwjpv8Aa3n/AGxYAspuARISOMtnnJABNbNFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB//9k=)

**PRACTICAL - 9**

**AIM: Write a program for game Tic-Tac-Toe using MINIMAX Algorithm in python.**

**PROGRAM:**

**import numpy as np**

**from math import inf as infinity**

**game\_state = [[' ',' ',' '],**

**[' ',' ',' '],**

**[' ',' ',' ']]**

**players = ['X','O']**

**def play\_move(state, player, block\_num):**

**if state[int((block\_num-1)/3)][(block\_num-1)%3] is ' ':**

**state[int((block\_num-1)/3)][(block\_num-1)%3] = player**

**else:**

**block\_num = int(input("Block is not empty, ya blockhead! Choose again: "))**

**play\_move(state, player, block\_num)**

**def copy\_game\_state(state):**

**new\_state = [[' ',' ',' '],[' ',' ',' '],[' ',' ',' ']]**

**for i in range(3):**

**for j in range(3):**

**new\_state[i][j] = state[i][j]**

**return new\_state**

**def check\_current\_state(game\_state):**

**# Check if draw**

**draw\_flag = 0**

**for i in range(3):**

**for j in range(3):**

**if game\_state[i][j] is ' ':**

**draw\_flag = 1**

**if draw\_flag is 0:**

**return None, "Draw"**

**# Check horizontals**

**if (game\_state[0][0] == game\_state[0][1] and game\_state[0][1] == game\_state[0][2] and game\_state[0][0] is not ' '):**

**return game\_state[0][0], "Done"**

**if (game\_state[1][0] == game\_state[1][1] and game\_state[1][1] == game\_state[1][2] and game\_state[1][0] is not ' '):**

**return game\_state[1][0], "Done"**

**if (game\_state[2][0] == game\_state[2][1] and game\_state[2][1] == game\_state[2][2] and game\_state[2][0] is not ' '):**

**return game\_state[2][0], "Done"**

**# Check verticals**

**if (game\_state[0][0] == game\_state[1][0] and game\_state[1][0] == game\_state[2][0] and game\_state[0][0] is not ' '):**

**return game\_state[0][0], "Done"**

**if (game\_state[0][1] == game\_state[1][1] and game\_state[1][1] == game\_state[2][1] and game\_state[0][1] is not ' '):**

**return game\_state[0][1], "Done"**

**if (game\_state[0][2] == game\_state[1][2] and game\_state[1][2] == game\_state[2][2] and game\_state[0][2] is not ' '):**

**return game\_state[0][2], "Done"**

**# Check diagonals**

**if (game\_state[0][0] == game\_state[1][1] and game\_state[1][1] == game\_state[2][2] and game\_state[0][0] is not ' '):**

**return game\_state[1][1], "Done"**

**if (game\_state[2][0] == game\_state[1][1] and game\_state[1][1] == game\_state[0][2] and game\_state[2][0] is not ' '):**

**return game\_state[1][1], "Done"**

**return None, "Not Done"**

**def print\_board(game\_state):**

**print('----------------')**

**print('| ' + str(game\_state[0][0]) + ' || ' + str(game\_state[0][1]) + ' || ' + str(game\_state[0][2]) + ' |')**

**print('----------------')**

**print('| ' + str(game\_state[1][0]) + ' || ' + str(game\_state[1][1]) + ' || ' + str(game\_state[1][2]) + ' |')**

**print('----------------')**

**print('| ' + str(game\_state[2][0]) + ' || ' + str(game\_state[2][1]) + ' || ' + str(game\_state[2][2]) + ' |')**

**print('----------------')**

**def getBestMove(state, player):**

**'''**

**Minimax Algorithm**

**'''**

**winner\_loser , done = check\_current\_state(state)**

**if done == "Done" and winner\_loser == 'O': # If AI won**

**return 1**

**elif done == "Done" and winner\_loser == 'X': # If Human won**

**return -1**

**elif done == "Draw": # Draw condition**

**return 0**

**moves = []**

**empty\_cells = []**

**for i in range(3):**

**for j in range(3):**

**if state[i][j] is ' ':**

**empty\_cells.append(i\*3 + (j+1))**

**for empty\_cell in empty\_cells:**

**move = {}**

**move['index'] = empty\_cell**

**new\_state = copy\_game\_state(state)**

**play\_move(new\_state, player, empty\_cell)**

**if player == 'O': # If AI**

**result = getBestMove(new\_state, 'X') # make more depth tree for human**

**move['score'] = result**

**else:**

**result = getBestMove(new\_state, 'O') # make more depth tree for AI**

**move['score'] = result**

**moves.append(move)**

**# Find best move**

**best\_move = None**

**if player == 'O': # If AI player**

**best = -infinity**

**for move in moves:**

**if move['score'] > best:**

**best = move['score']**

**best\_move = move['index']**

**else:**

**best = infinity**

**for move in moves:**

**if move['score'] < best:**

**best = move['score']**

**best\_move = move['index']**

**return best\_move**

**# PLaying**

**play\_again = 'Y'**

**while play\_again == 'Y' or play\_again == 'y':**

**game\_state = [[' ',' ',' '],**

**[' ',' ',' '],**

**[' ',' ',' ']]**

**current\_state = "Not Done"**

**print("\nNew Game!")**

**print\_board(game\_state)**

**player\_choice = input("Choose which player goes first - X (You - the petty human) or O(The mighty AI): ")**

**winner = None**

**if player\_choice == 'X' or player\_choice == 'x':**

**current\_player\_idx = 0**

**else:**

**current\_player\_idx = 1**

**while current\_state == "Not Done":**

**if current\_player\_idx == 0: # Human's turn**

**block\_choice = int(input("Oye Human, your turn! Choose where to place (1 to 9): "))**

**play\_move(game\_state ,players[current\_player\_idx], block\_choice)**

**else: # AI's turn**

**block\_choice = getBestMove(game\_state, players[current\_player\_idx])**

**play\_move(game\_state ,players[current\_player\_idx], block\_choice)**

**print("AI plays move: " + str(block\_choice))**

**print\_board(game\_state)**

**winner, current\_state = check\_current\_state(game\_state)**

**if winner is not None:**

**print(str(winner) + " won!")**

**else:**

**current\_player\_idx = (current\_player\_idx + 1)%2**

**if current\_state is "Draw":**

**print("Draw!")**

**play\_again = input('Wanna try again?(Y/N) : ')**

**if play\_again == 'N':**

**print('Suit yourself!')**

**OUTPUT:**

