**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Analysis and Design of Algorithms**

***Submitted by***

**PRANAV KAUSHIK N G (1BM20CS0109)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “**Analysis and Design of Algorithms**” carried out by **PRANAV KAUSHIK N G(1BM20CS109)** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Analysis and Design of Algorithms - (19CS4PCADA)** work prescribed for the said degree.

Name of the Lab-In charge: REKHA G S                **Dr. Jyothi S Nayak**

Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1** | Write a recursive program to Solve  **a)** Towers-of-Hanoi problem **b)** To find GCD | **5** |
| **2** | Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N. | **7** |
| **3** | Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort. | **13** |
| **4** | Write program to do the following:  **a)** Print all the nodes reachable from a given starting node in a digraph using BFS method.  **b)** Check whether a given graph is connected or not using DFS method. | **14** |
| **5** | Sort a given set of N integer elements using Insertion Sort technique and compute its time taken. | **18** |
| **6** | Write program to obtain the Topological ordering of vertices in a given digraph. | **22** |
| **7** | Implement Johnson Trotter algorithm to generate permutations. | **26** |
| **8** | Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort. | **28** |
| **9** | Sort a given set of N integer elements using Quick Sort technique and compute its time taken. | **33** |
| **10** | Sort a given set of N integer elements using Heap Sort technique and compute its time taken. | **33** |
| **11** | Implement Warshall’s algorithm using dynamic programming | **37** |
| **12** | Implement 0/1 Knapsack problem using dynamic programming. | **38** |
| **13** | Implement All Pair Shortest paths problem using Floyd’s algorithm. | **39** |
| **14** | Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm. | **40** |
| **15** | Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm. | **42** |
| **16** | From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm. | **45** |
| **17** | 1. Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t have a solution. | **48** |
| **18** | Implement “N-Queens Problem” using Backtracking. | **50** |

**Course Outcome**

|  |  |
| --- | --- |
| **CO1** | Ability to **analyze** time complexity of Recursive and Non-Recursive algorithms using asymptotic notations. |
| **CO2** | Ability to **design** efficient algorithms using various design techniques. |
| **CO3** | Ability to **apply** the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete |
| **CO4** | Ability to **conduct** practical experiments to solve problems using an appropriate designing method and find time efficiency. |

**Program 1:** Write a recursive program to Solve **a)** Towers-of-Hanoi problem **b)** To find GCD

**Tower of Hanoi**

#include <stdio.h>

#include<time.h>

void towerOfHanoi(int n, char from\_rod, char to\_rod, char aux\_rod)

{

if (n == 1)

{

printf("\n Move disk 1 from %c to %c", from\_rod, to\_rod);

return;

}

towerOfHanoi(n-1, from\_rod, aux\_rod, to\_rod);

printf("\n Move disk %d from %c to %c", n, from\_rod, to\_rod);

towerOfHanoi(n-1, aux\_rod, to\_rod, from\_rod);

}

void main()

{

int n;

time\_t start,end;

printf("Enter the number of discs");

scanf("%d",&n);

start=time(NULL);

towerOfHanoi(n,'A','C','B');

end=time(NULL);

printf("\n Time is %fs",difftime(end,start)); }

**OUTPUT:**

![Text

Description automatically generated with medium confidence](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAkACQAAD/4RDoRXhpZgAATU0AKgAAAAgABAE7AAIAAAAKAAAISodpAAQAAAABAAAIVJydAAEAAAAUAAAQzOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHlhc2Fzd2luaQAABZADAAIAAAAUAAAQopAEAAIAAAAUAAAQtpKRAAIAAAADMDQAAJKSAAIAAAADMDQAAOocAAcAAAgMAAAIlgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADIwMjI6MDc6MjIgMTE6MDU6NDUAMjAyMjowNzoyMiAxMTowNTo0NQAAAHkAYQBzAGEAcwB3AGkAbgBpAAAA/+ELHGh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMjItMDctMjJUMTE6MDU6NDUuMDM2PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPnlhc2Fzd2luaTwvcmRmOmxpPjwvcmRmOlNlcT4NCgkJCTwvZGM6Y3JlYXRvcj48L3JkZjpEZXNjcmlwdGlvbj48L3JkZjpSREY+PC94OnhtcG1ldGE+DQogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgIDw/eHBhY2tldCBlbmQ9J3cnPz7/2wBDAAcFBQYFBAcGBQYIBwcIChELCgkJChUPEAwRGBUaGRgVGBcbHichGx0lHRcYIi4iJSgpKywrGiAvMy8qMicqKyr/2wBDAQcICAoJChQLCxQqHBgcKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKir/wAARCAKKAgADASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDi9P8Ainrfwykd9BtbC4OoDEv22N227Om3a6/3j1z2q/8A8NV+N/8AoFeH/wDwHn/+PVyV/wCHp/E2tWFjDNHbRrHPPcXMudlvCiqzyNjnAA6dzgVUtPB3hzXLyXTfC3iq4vNVORawXul/Zo7xgM7UkEr4Y4OA6rk4GRmuzHf7xL+uhyYL+BH+up3H/DVfjf8A6BXh/wD8B5//AI9R/wANV+N/+gV4f/8AAef/AOPVw3hjwVo+ueE9S1vU/EkmkDSpoku45NP8wESbgnlkSZdsp90qoGeuM1T1vwrp9hoem67pGsS6jpF5O1tK8lmIZ7aVQGZGj8xgflYEEPg9OK43o7Py/Hb8zrPRf+Gq/G//AECvD/8A4Dz/APx6j/hqvxv/ANArw/8A+A8//wAern/EmjeB4PBfhGf+0tUtvtFpcN9og0OEyXOLhxukH2kYI+6OW4A5HQcp4f8ADFvr2oahIdTNjomnIZrnUbi3+ZIywVP3SscuxIAUN688Zoe7XZv8Aen9dz0v/hqvxv8A9Arw/wD+A8//AMeo/wCGq/G//QK8P/8AgPP/APHq4D/hEtF1XTb2bwl4guNQvLGJ7iSyvdPFq8kKcu8ZWSRWwOdpKnAJGcYqTwj4I07xbomp3Ca+dOvdMgFxOl5aqtsIzIq584SFs4bOPL56DJNAdvu+Z3f/AA1X43/6BXh//wAB5/8A49R/w1X43/6BXh//AMB5/wD49Xl+u6X4ds7NJdA8Ry6pKJjHJDPpzWx24yJEO5gy5yOSrf7OOawqAPbf+Gq/G/8A0CvD/wD4Dz//AB6j/hqvxv8A9Arw/wD+A8//AMerj/hHpmgap4uEOuvNJJ9mujHbfYI54XAtpDuZmkXDLjIG08gcjqOZ1S38Ow2qtoeq6pd3G8ZS70yO3UL6hlnkJOccY/Gm9Gk+1/xa/QN1c9W/4ar8b/8AQK8P/wDgPP8A/HqP+Gq/G/8A0CvD/wD4Dz//AB6s3xvb6VrHh/wXqvi/xReRXVxoyR7YrU3s74lkzJIWkTC8gDkk4PHFcm/w8ux41Ghx39s9o1qL8amQwiFmU8zziv3h8v8AD1zx70no2vNr7v6/qwk04qS6pP77f52O+/4ar8b/APQK8P8A/gPP/wDHqP8Ahqvxv/0CvD//AIDz/wDx6uHtPB3hzXLyXTfC3iq4vNVORawXul/Zo7xgM7UkEr4Y4OA6rk4GRmuKIKsQwwRwQe1Az23/AIar8b/9Arw//wCA8/8A8eo/4ar8b/8AQK8P/wDgPP8A/Hq8UiEZmQTMyRlhvZF3EDuQMjJ9sivQtV+H/hPRLnTItT8b3Ef9qW0N1AE0fe0MUqblaYCbCdcYUue+MU7N/kB1P/DVfjf/AKBXh/8A8B5//j1H/DVfjf8A6BXh/wD8B5//AI9XDWXw9DeJPEWgapqT2uo6NbT3CeTbiaKcQoWILb1KZAGDtbrzjFZ+geGtP13QNYuV1O6g1DS7N702xslaGSNWRceb5oIbL9NmOOtTdWv5X+Q7a287fPT/ADR6T/w1X43/AOgV4f8A/Aef/wCPUf8ADVfjf/oFeH//AAHn/wDj1Ylr8JdDv20K2s/GUg1DxBaNc2NtPpW3AAb/AFrLKwQEoQCN30FHh/w7D4d8ReG/EXg/xTPf2kmvx6VcSx27WjbtyMQPnO+NlJ649CKrrZ/10/Mi/u8y9f1/I2/+Gq/G/wD0CvD/AP4Dz/8Ax6j/AIar8b/9Arw//wCA8/8A8ern/FvhXwqfiHqtnqXjUWuoXN/O7+Xpxltbd2kJVJJvMBzyASqMFOeeKxLf4evZT6u3jDURollpM4tZZkgNw807AlUiQFd2VBbJIGMHvURd0mW1Z2O7/wCGq/G//QK8P/8AgPP/APHqP+Gq/G//AECvD/8A4Dz/APx6vOr7wrps3hy41nwtrUupxWJX7dbXVl9lnt1ZtqvtDurruwCQ2QSMjnNaOieA9H1vwbN4g/4Sc6fHYzxQ363tkFWPerH90VkZpT8uAu1Sc9gCarv5C7I7T/hqvxv/ANArw/8A+A8//wAeo/4ar8b/APQK8P8A/gPP/wDHq8o12w0Sz+zP4f1uXVI5VbzVnsjbSQkHAyNzqQRyCGPuBWTQB7b/AMNV+N/+gV4f/wDAef8A+PUf8NV+N/8AoFeH/wDwHn/+PV5ppPhm1m0B9d8Q6k+maaZjb25htvtE1zKACwSMugwoIyxYDkAZNLqnhi0Tw8dd8Oam+p6fFMsF0J7b7PPbOwJXegdxtbacMGPIIODQ9NwWp6V/w1X43/6BXh//AMB5/wD49R/w1X43/wCgV4f/APAef/49VHQLvRfDvwbm1HRvEeu6XcXesLbXF/Z6XGtxxBv8lSLkER5ydwYEngrivNPEWqz6xrMlxPrGpayqqEiu9Tz5zKOcEF3wASeAx/Wh72Baq56z/wANV+N/+gV4f/8AAef/AOPUf8NV+N/+gV4f/wDAef8A+PV4lXVweE9LsdEsdQ8Wa3Ppj6ivm2lpaWAuZmhyV81w0kYVSQccknGcYoA9C/4ar8b/APQK8P8A/gPP/wDHqP8Ahqvxv/0CvD//AIDz/wDx6sP4oJ5fw78CKuutrsH+nCC+Yvlow8QVSr8qVA27e2OOKx9WtYk+EkUmi+IL280ddb2tY3mmRW7LcGDJkDrLISNoAwSB3xRok3/W9hLWx2n/AA1X43/6BXh//wAB5/8A49R/w1X43/6BXh//AMB5/wD49Xmng7woPFN1fefffYbPTrU3VzKkDTybAwUBI1ILHLDuABkkip77w94Z/wCEWutW0XxU9zc28scf9m3lgttO4YnLriVwQMdievOOMj0V3/WthrU9E/4ar8b/APQK8P8A/gPP/wDHqP8Ahqvxv/0CvD//AIDz/wDx6uB/4RDSNK0iwu/F2vz6dc6jEtxb2VlYC6kWBvuySEyRqucZABJxzgU698D6fLrvh/TPC/ie01ltbYIHMfkfZmMmwCRSSVOOcflngl2d+Xzt8xXVr/M7z/hqvxv/ANArw/8A+A8//wAeo/4ar8b/APQK8P8A/gPP/wDHq45fBfhF9Zu9KbxrcWl3apIS1/o/2eKVkUttVjNkEkYG9Vzn1wDwdTdMpprc9t/4ar8b/wDQK8P/APgPP/8AHqP+Gq/G/wD0CvD/AP4Dz/8Ax6vEq9r+EsaeDNa8NrIo/tzxPOpwRza2AyQfZpWUf8AX3qkr/wBf1/wxEpcqb7a/1/W47/hqvxv/ANArw/8A+A8//wAeo/4ar8b/APQK8P8A/gPP/wDHq8Wuf+Pub/fb+dR1Kd1c0lHlk49j23/hqvxv/wBArw//AOA8/wD8eo/4ar8b/wDQK8P/APgPP/8AHq8Sopkntv8Aw1X43/6BXh//AMB5/wD49R/w1X43/wCgV4f/APAef/49XiVFAHtv/DVfjf8A6BXh/wD8B5//AI9R/wANV+N/+gV4f/8AAef/AOPV4lRQB7b/AMNV+N/+gV4f/wDAef8A+PUf8NV+N/8AoFeH/wDwHn/+PV4lRQB7b/w1X43/AOgV4f8A/Aef/wCPUf8ADVfjf/oFeH//AAHn/wDj1eJUUAe2/wDDVfjf/oFeH/8AwHn/APj1H/DVfjf/AKBXh/8A8B5//j1eJUUAe2/8NV+N/wDoFeH/APwHn/8Aj1H/AA1X43/6BXh//wAB5/8A49XiVFAHtv8Aw1X43/6BXh//AMB5/wD49R/w1X43/wCgV4f/APAef/49XiVFAHtv/DVfjf8A6BXh/wD8B5//AI9R/wANV+N/+gV4f/8AAef/AOPV4lRQB6jZ+K18H+KLK9mNyttcW1xaXElm+yeNJFUb42yMOpww57Y4ofxBeR3UjH44apLZAZXyTqBuW9vLYKgP/bTHvXL+Nf8Aly/7af8AstctXZjta8v66HHgtKC/rqdzotzo8fws8RWN14hs4tR1K4t54bWSG4MhEJlyGZYigZt4x8xHqRUNzc6T/wAKctdNTXLOTU49UkvWsVin3hHjSPbuMYTcCpJ+bGOhJ4rjKK43rfzt+Fv8kdt3e/r+J3zy+HfE/gjwzY3viW30S70dJ7e4ju7SeTzEeVpA6GJGB+9jDbenWpPCnijSfCGs6zpOm6/qkWlarbJB/bNpC1vPBKhDLKqK+4pncCuQSp9eK89op395vvf8dxHo+oa9ff2ffpd/GPUdShaJ0itbZ7+RrjII2uJRGiqeh+ZuCeDVbwf/AGHZ+D/ENvqPirTLO51mxSCKCSC7ZoWWdHO8pAy4wh+6W6iuBopDvqn2d/yf6F3VbG30+7ENpqtpqiFAxntEmVAcn5cSxo2ePTHPWqVFFAjp/h1rdj4f8cWl9q8jxWZinglljTeYxLC8e/b1IBfJA5wKTUNB8M2Gk3E0XjCHU73OLe2sdPnCtz96R5VTaMZ4UNzXM0UPXX5fjf8AVgd145m0XUNM8Kw6Z4jsLx7DT0sLkRw3K+W3mOxk+eJcoAw6Zb/ZroJfG2keHfGujalpmtjUbQ6DHo98+mLPDPb7YgjSRtIic7sMpB/hOduRXktFO99+9/vv/mxJWSXZJfda35I9RfxBeR3UjH44apLZAZXyTqBuW9vLYKgP/bTHvXl7ndIx3FsnO5up9zSUVJV9LD4I1luI45JkgR3CtLIGKoCfvHaCcDrwCfQGu58dzaLrGuaCdN8SafPDFplrY3E4hulWB4Y1VmYNCGKk5xtDHjkCuDoq1Jq3k7/df/MXc9UvfFHh+P4z6zqkesRXGj63aXNub2G3m/0bzoigLI6KxIIBIUHg8EniqfhmHwnoWl+ILe68Z2c17qmmSWkDQ2V19nTLo3zsYg+Tt4AQjrk9BXm9FRypR5fK3y1/zGnZ387/AD0/yR7BYeJPDOl+MvAF+3iO0uLbRtPNnfNDb3P7pv3p3YaIFl/eKOMnOeMc1m+E9X8P6f4c07S77xFZwyaf4tj1JpTBcsktukYG9MRE8lcAMAeRkAc15jRVX1b7u/43/MjlVkuyS+5W/I9G8Qad4C1zxlfazb+NPsum3dzLcS2sunztcqS5JWPC7GDckFmUgEZGc1q23xUTWL3xDaS6zqPhOHU7uO6stQsXkLQGOPyhHMIyGZWRVzjOGGcGvJKKlKyt0Kerud5r2t3s3hy6g1D4p3+vNIVCafC97JFKNwP7wziMADGeFbkD6ifTotAX4X3uhzeMtIhvb3ULe8UNb3pWNUjkVlYi3PzZcdMjg8155RT6Ndx31T7X/FWLF/bRWd9JBb30F/GmNtzbLII34zwJFVvblR0/Gq9FFAjsNP1DSNd8E2/h7WdTXR7nTLmW4sruaGSSGVZQu+NxGrOpBQEMFI5IIHBp17qGj+H/AAPe6Bo+qLrN5q08Mt5cwwSRwQpFuKonmKrsxZsklQAAAM9a42iiXvfh+ALT+u56MkXh5/hUvh1vGmjpenWP7QJa3vtip5GzaSLf72fbHvXAXtvHa3ssEF5DexocLcQBwknuA6q35qKgopW1uHSwV3N7d6H400TRm1HXoND1TSrNNOkS7t5pIZ4UJKOjRI5DANgqQAcZB7Vw1FPoHU9F8Y3HhW68E+FNF8O+I0uW0u4uYrqS6tZoifOdW84AIw8sbTxkvgj5euAW3h4fDRvDv/Cb6L9rOri+3/Zr/wAvy/JKYz9mzuyfTGO9edUUPVW/ruK2tzr/AAnDoVpNqkt54vudE1O1ZF0u/sI5jHLksHY7VEgXAGD8p+b7p6Vr674jtpvAd/p2veKx4w1WaeJtPm2XDmxUHMjebOiP8w+XYAR3rzmik1dWH1ud3qk3h7xnYaReT6/b6DqVnZQ2F3Be288kcqxLsWWNokfqoGVYDkHk1Vj0/wABReKtMsn1zVJNMVZDqOqpB5YZtpKeTFtLAAgAluTnovWuOopvV3+YdLHrWl+KbXTryZ/EnxGk8VaF5DodHniu5HucrhFKzJ5cZBwdwckY4zXktFFLrcOlgr1XwH8SLmbxvZat4x1nQ7WCzeMyTXGhRPcTIF2gJJFbs4IAAyWXjGDXlVFUnYmUVJWNHX5hceILyVbmzug8hImsbfyIX91j2JtHttFZ1FFSlZWLk+ZtsKKKKYgooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOt8dW80H2Dz4pI93mY3qRn7vrXJV1PjX/ly/wC2n/stctXXjf48vl+RyYP+BH5/mFFFFch1hRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHU+Nf+XL/tp/7LXLV1PjX/ly/wC2n/stctXZjf8AeJfL8jkwf8CPz/MKK0Lnw/rNlpMGqXmkX9vp9wQIbuW2dYZcgkbXIwcgHoe1Mi0XVZ9Hm1eHTLyTTbdwk16luxhjYkAKz42g/MvBPcetcZ1lKiiigAooooAKKKKACiitW38Napd+G7zXraKGTT7JlW5YXUXmRbmCqTFu34JIAO3HX0NAGVRWhpWhalraXz6Xbeeun2j3lyd6r5cKkBm+YjOMjgZPtWfRsAUUVq6d4Z1TVtHvtT0+OCW309DJcj7XEsiIMAv5RYOVywGQpGTijpcPIyqKKKACiremaVqGtXy2ekWU97cuCRFBGXbA6nA6Adz0FWvEHhrVfC95Da65bLbzTwLcRBZklDRkkA5RiOqnjrxQBlUUUUAFFb2meC9b1bR/7Vt4LaDTzKYVub6+gtI3cDJVTM67iPbNZN9ZS6dfSWtw0DyRkBmt7hJ0PGeHQlW69iaAK9FFFABRRRQAUUUdaACit278DeLbCxkvL7wvrVtaxLvknm0+VERfUsVwB71hUAFFFTR2V1LZzXcVtM9tAyrLMsZKRls7QzdATg4z1waAIaKKKACiiigAoro1+HXjZ1DL4O19lIyCNLmwR/3zWDd2lzYXclrfW8ttcRNtkhmQo6H0IPINAEVFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAdT41/5cv8Atp/7LXLV1PjX/ly/7af+y1y1dmN/3iXy/I5MH/Aj8/zPbPCksGofCTQfCV+yJb+IrjULeGRzgRXSGJoGz/vZX6PU+vwfZvhbqvgWxKE6LcaZbzsDw97M0rTHPoGKp/wCvOL7xVot74A03w8dFv0uNOaWWK7GpIUaSXbvLR+Rnb8nChgRnkmm2nivTdP8A6hoNlpd6l9qEtvNLfPqClEeFmIKxCIEAh2HLnnBzxg8r5Xe/wDd/Br9L/h2OmN4wst/e+V2/wDgfj3L+u3Hhrwp4huNAi8LWurrp0ptrm+vLq5Sa4kU4dkEcioi5B2gqxxgnNbOlaB4X0z4h2Wi6lpkF/pfiC0iudNuL6adZLVpoyYlfypEDDzCFb1HIIrn7vxd4d1u+XVPEnhe4utVb5rqWz1MW8N2/wDfePymIY9TsZcnJ4zWFr/iK88Qa4dTuBHbsqpHBDbgqlvGgCoiDsFAH86ldL/P/gfO1tCn1t8v6/M67QrHS7DQ/FmqeJ/CmnyLp032e2hkmukK3btgQArMMoiq7HOW4Hzc1fg8IWGjeE9GmfRtD1jUdUtVvpJdX8QR2SwRsxCokXnxOeFyXORk4HQ1geOPiG/jKwsraPTI9NCSNdXpjl3C8umVVaYjaNvCcDnG4881Ba+L9Ou/Ddro3izRJNUWwBSxu7W9+zXECFtxjLFHV1ySQCuRk4NGtvu/4P4/gD3+/wDr7vxJvHmhaRp9tpuo6QbG0lvBIl1pVpqsV+LZkxhlkR2OxwcgMcghuSMVxta2t6hot5HbRaFobaYkIPmSzXjXEs5OPvHCoMY/hQdTnNZNIbOv8P6Vpdh4Lu/Fmu2P9qRrerp9nYmZokaUoXZ5GQhtoXGApBJPUYrpNO1LQ9S+DXjWTS9DXRr6NLFbhLe4kkgmX7SMMokZmUjkEbiDx0rjfD/ieHTNLvNH1nTv7V0e8dZXthOYXilUELLG+G2tgkHKkEcEdK3E8eeHrTwTrHhnSvCc9tb6pFHvu31MSXBljfejM3lBSgOPkVVzz82SCG9/6/rv/W09P6/r+vvg8FwaRrWj65Y3+gWT3Fjo11eRags1ws3mJgrkCXyyBnps7CuZ0CHTbjxFp8OvXEltpklwi3U0a5ZI8jcR+Hsfoa6Pw14r8M+HrW7H/CP6tc3F9p8lhcyf2xGiFZAAzIv2YlTxxktj3rP03XPD+j+LbPU7Tw7LeafAjCXTtUvEuBMxVlyWEKAAZBxtJyvX0d7Tv/W7C14tf10O7GiaB9u1Av4I0m/0uCCR4pdB8RG7ukAQ7ZHj+0EsobBb5EwMntg4vwi0m412TxZpdk0Sz3WgvGjTOERSZ4eWY9AOtUtL8aeHfDurnWvDnhe6tdURHFsbjVvOt7dmUruEflK7YBOA0hHrmsvw74ms9A0LXLX+zp7i+1azayFx9rCRwxlkbPl+WSzZTrvAwenekrLmv/K1+DX6l31Xr/kWvGMmh6XI3h7S9AgS7sAILrVZTcpNPMpw7CJpCiKemCue/wAucDkq6nXvFWm+ItAsk1DSrttftYRC2rG+XFwobjzIvKyxC/KDvz0yWwBXLUkmtxBXcfEv/U+DP+xYtP8A0OWuS0yTTor0NrFrdXVrg5jtblYHJ7He0bj8NtdR4n8W+G/EdjYxr4f1W2uNO05NPtZDrEboFQsVZ1+zAscsc4Zc+1N7f15kr4r/ANdDjKdGwSRWZFkCkEo2cN7HBB/I02nwmNZkM6M8QYF1RgrMueQCQcH3wfoaAex6n4l8QabH8L/BMz+ENFljlF/sgea92Q4nAO0i4DHPU7i3tgcVzXw4h0nWfiLYaXq+h2d5Z6pdrEY3luE+zqSf9WUlB9vnLdKs6h4w8Jal4d0jRp/DWtLbaR53kMmuRB28197bibTB5HGAPxrN8G+JNG8K+JodauNHvr6azuVns401FIVUAn5ZMwsX7cjb9KcLKSuTO7g1Hcm1jVfB8C3umaV4WeRER44NTuL6Rbnzc8SFATFs6jZtJx/HnmrWtw6H4IltdHn8PWmt6j9miuL25vp7hVVpUWQRxrDImAqsAWbJJz0Fc/rd74fvE36LpWp2Vw0pd3u9SjuEKnOQFWCMg5xzk/TvWq/izRtYs7MeLNBur++s4kgW9stRFq00SDCLKGikDEDA3DacAZzjNJX5ddzWdud8u2v/AAPwOy8N+F/CU99Bqt3oZu9F1TRbvUEtJLqVZrSW23B40dWAKkgYLhjjHcHLrSy8DXFh4PvX8FRiTxNfPZyQrqVx5dsiTCMunz7i5Ei/eJX5enNc9p3xLsrbVJpbrQJPsCaZLpdhYWd+IVtYZQwkJZo3Lud2d3HPUEYAoweObODSfC9qukTmbw5qL3cMpvRtmR5VkMbL5eQ3yKNwOOvy88C+JX2/4L/SxGtn3/8AtV+tznfEGnJpHiXU9Nhdnjs7uW3Vm6sEcqCfyrPq/rupDWfEWpaokJgW9u5bgRF95jDuW27sDOM4zgZ9BTdHubG01i2uNXspL+zifdLbRTiEy46DeVbAzjPHIyOOoUFolIqT1bR7doOjeIfDVr4N0SDw1q15pN9C9xrslvYyvHILtfL2khSMxxbT9a8W8R6JceG/E2o6NegiaxuHhYkY3YOA30IwR7GtLxb4k07xT40l1xtOvoIrqUSXVu9+kjHn7sbiFQgC4UZVsY79KTxx4ntfFuuQajaadNYslnFbzefdCeSdoxtEjMEQZKhQeOoz3oer5v68vu2EtFy/1fr9/wChzlem+F9ZsYPg14m83w1pdx9nurBZPNkuh9pJM2GfbMMEY42bRycg8V5lXSeGvFFppGi6to2saU+pabqnlM6w3X2eWKSMsUdX2uP42yCpB4qvsNd7fg0/0BaO4aVYxeOPGul6Vp+nWWhreSpAwtWmdFGclz5sjsTjsCBwPrVx9d8HpqhtB4OibSlfy/tRvbj7cV6eZnzPK398eXjt71UvPE+n2t/pt14O0T+wp9PlEy3L3jXM0jgggsxCpgY6BB1Oc1cPijwk+pf2q/gtzfE+YYBqZFkZeu7yfK37c87PMx26UK2l/wCv6+Qa6/L9f+AY/i/w+fC3i/UtFM3nizmKJLjG9TypI7HBGR61jVc1fVbvXNYu9U1KXzbq7laWV8YyxPYdh6DsKp1EbqKvuN2voenarpNjqHw78AyXviTT9FKWtyALqO5ZyPtcnzL5UTjj3Iqn8YUd9b0e6jn/ALRszpUNvDrAcONSaLIeUkEkMCdpVjuG0Z61V1Dxd4T1TQ9H0u68N6yItIhkihePXIgzh5DIxbNqe7HGMcevWsrxT4qg12y0rTNL0waXpWkxuttbtOZ5C0jBpHeQgZJIHQADHAq56ybX8zf33/zBfp/kc5RRRSEFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB1PjX/ly/7af+y1y1dT41/5cv8Atp/7LXLV2Y3/AHiXy/I5MH/Aj8/zCivV7PwdoOvfBWC+stNS18T4uZ45YppT9rjtinmKUZyu4rIW+UD7nao73wZoekfA2XULqy83xM5tbpp2lkH2aC4d/LUIGC5KxEncCcSfTHG/du35fjsdcfeSa8/w3PLKK6t/AbWbw2+t+I9F0e/lVW+w3jzmWMMMqJDHEyRkgg4ZgRnnFSaT8ONR1LxDe6Fd6jp+lanZo0ptrwzM0yKhkLRmKNw42jcADkjoDQByFFdRoPglfEEeqyW/iTSLePS0eaaS4W6CtCpA81SsJ4JYAKcOT/DTbHwTLPoMGr6trek6Ha3TEWo1B5TJcAHBdUijdtoORuIA4NAHM0Vs+IvDF54bltzcTW15aXaGS1vrKXzILhQcNtbAOQeCpAIPUdKxqACitrw/4XuvEEd3cLc2lhYWShrm+vZCkUW7O1flBZmODhVBJweK7m38MR2PwX8U3hutA1q2j+ytaX1jCpmt5GnUOrM8aTJlccHjGcd6NhXPLKK7Pwjoxn0PW7uxvPD17cf2TcPNYX9vcPPbRqQTLE3l+WJBgbTvPDfXHL6Vpl3rWrWumabCZru7lWGGMHG5mOByen1p215UF1a5Uort7X4ZSahqFxp+m+LPDd5fW8bySW8VxPkhFLMFYxBXICn7hP5c0eA/Ku/DXjGzu7Oynjt9Gku4ZJbOJpYpRLEoZZSu9eCeAcc5xmpvo/JN/JX/AMiuV3S87HEUV0Wt+EG8P6Va3OoazpwvLmFJhpaif7TGrdN4MQRTjnBbkcjIIrnaYgora8K32i6brJvfEVhJqMMELvBaD7ks+P3Yk5B8vPJxknGMcmutt7mPxd4A8SX+uaJpOnx6cqSWOo2FjHaHz2dQLbEYAkBQsecsu3JND0V/6/r+twWrsecUUU6NQ8iqzrGGIBds4X3OAT+QoAbRXr7aTpHhr4c6BNp2u+EVudSluZJ9R1HSZ7vzwjhFWMSWr7FA65VST0JFcPo+gy+OvGD2Eer6PaXtzMIrcm2kghuW+6PLSGHCA4B+ZU6885o3dkK6SuzmKK6vUvAFzpFtcjUdc0O31G1iaWXSnuz9oABAK52+Xvwc+Xv3+2eKhh8Fsmm2l5reuaXoYvU822hvjO0skZ6SbYonKqecFsZ6jI5oKas7M5qiu20r4WavqviCXRxqOlW04tPttvLPO/k3cGCTJHIqFdoAOdxX06ggW4/hBfzR2U0Xifw21tqUvkWFwLuXbdTbipjQeXuyCBksAvzLzzQL+v1/LX0PPqKmu7WaxvZ7S7Qxz28jRSof4WU4I/MVDQmnqgegUV7Fo2g+E9Y8PaJZan4bsNM1vxMt0NOntrm6AgCLthdlkmYEvKCOmMV4/LE8MzxTKUkRirKwwVI4Ioejt/X9LqC1VxtFFd/4f8G6FqPw01rVrzXtLhvYJ7QRyyref6GHMm5HCQkMW2jBUOBt6judG+36u36h1scBRWle6QkWpwWOlajba3JPtVGsI5gC5OAmJY0Yt06Ajkc1uH4fsuof2W3ibQF1fOw6eZ5t4k/55+b5Xk7s8f6zGeM0AcjRU13aXFhezWl7E0NxBI0csTjDIwOCD7g1DRuGwUV3mtweH/ArxaJdeHYNc1hIYpr65vbqdIomdA/lRpC6cKGXLMxyc8AVl+MJvCNzZaPN4PtJrKZ4JG1G3lleQxS7ztUM3BXb0I7Yzzmh6AtVc5eiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOy+IFk9n/Z++SN9/mY2Z4xt9QPWuNrvfid/wAwv/tr/wCyVwVdeN/jy+X5HJg/4Efn+Z6nBf3nh/4c+ENV0fUNHbUNHvLu8ltW1O3aQRy+WFVovM3tuAYFANwGcgVK3iC6vvBPibxFrdzo4vNRv9NuoLCC/h3mOFnUosIcyKFBQYIzjnoCa8norm5n+X4NP9F+J1WXLy+v43/zZ6B4m8OJ4r8WXmvaLr2imw1a4e6zfapDbS2u87mSSJ2DkqSRlAwIGR1qvr/jGOy8faNqfhyc3CaBa2lpFc7SguvJUBmweQrcjB/hrh6KlaWt01Kb5r36np3jmfw94f8AC91Y+EdQtrxfE18L6UW8gY2tooDRW7gH5WDuxIOD8gzVpNdl8ReDdE/sK98MQX+k2S2N1Ya3aWIdwrEpJFNcphgQ3K7xgg4HOa8nooskrf1pp+Qtb3/rXU7Dxreau2m6ZZatq+gXSRNJJHZaLFbqtszbdxc26CMlsDozfd7d+PoooHfodt4emtNd+Hd74Ta+tNOv01FdStpL2YRRXOIjG0RkbCow4ILEA8jIrc0zQrbQ/hP4ws7rxFocuq6hDayx6fb6nDIQkU29vnDbGfGcIhZuOmSAfLaKOtybaWPR/h/oc1nb63c3mo6Hbx6joNzb2wl1yzR2kkVdilDLuQn/AGgMd8Vg6H4cltfHGnWGo+INP0RmzMuqW9/Dcx2xVWZSXikIB3KBjcDyOOmeWoo+1df1v/mFtGj160LXOqXTfEO/8G3+iNDI095YtY/a5G2/I0XkATl92PvjGM7qxvhVqp8MWnibXzcaYjppbw2sN5cRFpZ/NidQIS29hhScgYyOua86ooV1e3a35/5j7XO18aaTa6lCvi/Sb+zMOpAz3WnzapDJd2spfDLsL+Y6knIONwH3gMZPFUUUkktEF29zo/A/h3TvEmvtb61q1vpllBC08jS3EULz4IHlRtKypvbPG4gAZPOMV0Xi601PWLF/9K8OaPoWjwM1hpdtrttOSMgEBYpGaSZs5LEDODyOBXnVFOWqt/XqC0CnwxNPMkSFQ0jBQXcIoJOOWJAA9ycCmUUAen+IPDtxffDvwhplrqfh97zTRe/aov8AhILEeX5kwZOTNg5HPBOO9ZXwrsDb/EjStRu73TLS00y/RrmW61O3hAAJyV3uDIOOqZrhaKcXyu6JlHmi4vqbniHw/e6XJJd3Vzpk0c07Bfsmq2105JyclYpGYDjqRj866TxJpyeOru017RdU0mIy2lvb3NlfajDaSWskUSxnAlZQ6HaGBXPXBAIrz+iktFY0lJym59Xf8dT2nw/d6ahg0G31nS5V0fw7qFnLfS30VvFNc3O9ljiMrLvUEgbhxkk8AgnIsri0i8O/DR5NR08f2VrUwvU+3Rb4A1zGwYruzswjHfjbgdeRny2ihXUub+t2/wBSLKzXz/BL9DZ8ZSxT+Otelt5Y5opNSuGSSJw6OplYgqw4II5BHBqno+mtrGsW1glxbWvnvtM93OkMUY6lmdyAMAHvz0GSQKpUUopRSXYcru5614y8S2tl49sbbTNA8M38WnrDa6RfNq0j4iiO2NneG6WKP5gW+YLjOW9a5j4rQWI8eT3+m3FlLHqkKX0sVldR3CW80g/exl0ZlyHDHr0IrjKKOmu/9f0w0WwV3Pg77PqXw/8AFOgjULGz1C7ls7m2S+uUt0mERk3qJHIUN864BIzz6Vw1FV0a7/5p/mg2O30q3t/h14w0DWb7VdL1RobpZZ7XS7kXDQoMcl1Hl7uTgBjyKjfwGj6s0q+KdA/shmMo1A6lFv8ALznJt93nb8fw7OvGe9cZRQm1Z9g/X9L/AOZv+Otdt/E3jrVtYskZLe6uC0Qf7xUAAE+5AyfrWBRRURiopJdBt3dzv/ElpD8QL+PxFo+paZb3dzDFHf2F/fxWjQSogQsplZVdGCgjBJGcEVzfiXQ7DQJba1tNdtdXuyha7+xKWggbPyqsuf3hx1IAA6AmsSiqEtEFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAd78Tv+YX/ANtf/ZK4Ku9+J3/ML/7a/wDslcFXZjv94l8vyOTBfwI/P8zrJ/h5qcfw6j8aW95Y3emtKIpY4Wk86Bs4O5WQDAJAJUkZYVGnw/1VvhvL42lntIdOWcQRwyO/nTEtt3KoUrtzu5LD7je2e38Pa9Z6J8OfCcWtMw0bU7jU7HUdqlisT+T84A5yjBWGOfl4qxqWr6XrvgfxDp2l3Lf8I/plxo+n2kxUpuiUzb5SD0LMXbkA8jNcri9bf3fxaT/Nff5HTF+5zS8/wbt+X4eZ4xRXovjPxj4l8MeN9Q0XQdSvNF0zS7hre0sLSVo4fKU/KzIOJCwwxZs7t3pW4nic+Bfilpzw3E+lWGs6fbSa3Y2crQJBJPDh2Cr9xl3eYO6k4GKle9Zrrt+a/IqXutp9P6Z49RXrqXXib4aeG/F1zea3qMd9fag2m2hF1IPNkyHmu8Z5bYEAfr+8qSfVrTwn4C8LxaT4l1zQINRsftU82jadHJ9puBIwcPN9ojYlMBdmMAY65pXVr+n4q/5A73t6/hp+Z49RXZePPEWjeI7bTZ7Se+vtYiEiX2oXdhFam6Tjy9ypK4Z1+YFjgkbc5xXG0xhXR6Z4ZtNT8C63rqajNHd6OYTJaG1BjkWWQICJfMzkZJI2dhzzxteG7mbw98LtS8R6C/l60dTjsXukUGSzt2jZtyH+Auw27uvy4BGa2tO8Ua94l+CXjb+355NQFoLFIr64G6Xm4BMZkPLDjIBJx+ND6r0/r8SezPONKs9Ou0vjqmq/2c0Fo8tsPs7S/aZgRti+X7mcn5jwMVn16H8Odd1eTSPE2jyapevpkfhy9dLJrhzCjYByI87Qck847muT8KO0Xi3TJk0hta8m4WZtPRC5uFT5mXAByMA9iPXinpzJdP8AgsL+63/WyMiuj8O+GbPX9C1y6Ooz297pNm16Lf7IHjljDIuPM8wFWy/TYRgde1elaJ401DxH4lubbw14y1a9vL6GZbfSdf09BatmNsojRzERsFyVIRBkAHiuW+EUOnXEniyLW7mW009tBcXE0Kb3RPPhztHrSSbTX91v52f+RWl16nndFd58SNb8Qw3kfh+W7nh8OwQoNMtILt5Laa2HMcgJP7zPXJHB4AXGBwdJO4NWLemWlve3yw3uowabCQS1xOkjqPbEasSfTjHqRWx4y8L2/he60xbLUm1K31HTo7+OZrfySFdnAG3c39zOc9+lc5XcfEv/AFPgz/sWLT/0OWm9l6/5kr4reX+Rw9FFOjkeKRZInZHQhlZTgqR0INAzqrbwnpNr4R0/XvEut3NkmpyypZ21jp4uZHWM7Xdi0saqNxxjJJrm75LOO+kXTJ57i1BHly3EAhduO6B3A5z/ABGvTvEvjnxbB8L/AATdQeKNajuboX/nzJqEoeXbOAu5g2WwOBnpWN8Idb1WD4uaN5Op3kf9oX6C82XDj7SCxJEmD8/JJ5z1pxXNJImUuSLl2OCorsNb+I3iO7a/02G6Sz0iVXtxpdvCotkQtnhCCN+RnzPv553VreL/ABDrPg28sNF8I6nd6RpaafbXEb2EzQm8aSJXeZ2UguSxYckgBcADFSndX/rqaSjyzcO1/wALL9TzmivevCur3tldWHiCKQWuq6n4X1C7v4EULHcvD5ghnki+6xYDOSMEgnuaZYePPEj6T8N7iTU5HudZ1SWDUbhlUyXUSXKKsTtjLIBK/wAp459hT3fL/W7X6GfNo3/Wyf6nhFPgjWW4jjkmSBHcK0sgYqgJ+8doJwOvAJ9Aa1PFttDZ+NdbtrWNYoYdQuI40UYCqJGAA/AVkVMZcyTLkrOx2158N4tPvrSzvfGnhyCa8giuIBJ9sVWjkXcjFzb7VyD/ABEY74rmNc0W+8O65daTqsXlXdq+yRQwI9QQR1BBBB9DXpHi+/8ACVnrHhlvEmhanqTroOnGTyNUSGN18heNnklvykGfaue+LEUknjYaw9x58OuWcOo22YfKaKF12pGy5OCoTb1OQAe9VJWk10Ta+56fkw6fJM4mtS08P3V74Z1LXInhFrpssMUyMx3ky7tpUYwR8hzkjt1rLr03wv4t8R2fwa8TfZNf1SD7FdWEdr5V7Iv2dGM25UwflBwMgYzgU38Dfa34tL9QW55lRXXeGZrjxt8RtBh8Z6le6jb3F1HbvLe3LyFk3Z8sOxJAJOODxuqzL8RPGieJGt1vrlI1nMA0MA/ZAN23yPs33cfw4xn8eaEr2Xf+v1Jvv5W/X/I4iiuk+IemWGjfETW9P0gBbOC6ZY0VsiPuUz/sklfwrm6iMuaKZTVnY6i38DTDR7bUdb1vSdCju1D20N/JKZpoz0kEcUbsEODhmABxxxUHirwXqHhGPTpNQubK5j1KJpreSzn81WQNtDbsYweox2PODkVq/FpTJ48a9h+awvLK1lsZF+48PkIo2+wKkY7EGuXvE1YaXp76gt4LAq4sWnD+UQGy4izxjcedvc81T8u/9fMS217f18ijRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHe/E7/mF/9tf/AGSuCr0D4pRtH/Ze4qc+d91w39z0rz+uzG/7xL5fkcmD/gR+f5m9P411q48NQ6BM1g+nW6lYYzpltvjzjLLJ5e8Mdoy27JxyTTB4v1dfC83h5Hs49Nn2GVI9Pt0kkKNuUtKE8wkEnktnBI6EisSiuM6zprT4h+JLO3tIkurWY2SBLaa6063uJoFHQJLJGzqB2weO1c/eXlzqF7Ld388lzczMXkllYszsepJPWoaKN3cOljY17xZrniaCwh12/a7TTofItgyKuxMAclQNxwo+Y5JwOal0fxnruh6XNptjdxPYTOHezu7WK6hLf3gkqsoPuBmsKigDW1vxRq/iKO2i1W6V4LUEQW8MEcEUWcZ2xxqqjOB0HYVk0UUAaeh+ItV8OXMs2j3XkmaMxTI8ayxyof4XjcFWHsQa2Ln4meKrrR7jSZb62Gm3EPkvZR6dbJCFznKxrGFRs87lAbIBzwK5SryaHq0miyaxHpd42mRvse9W3cwo2QMF8bQckDGe4oFobGmfEHXtHtfs+nDSYUMBtnb+xLNnkjIwVdzEWcHHO4nPfNVYfGWs2viK21yxltLHUbVSsMtlYQW6rkEE7I0Ck4Y8kZ9+BWXaafeX6zmxtJ7kW0RnnMMRfyox1dsDhRkcnjmq9D3ux9DqZviR4nm+0MLy1hnuozHNdW+nW0NxIp+8DMkYkOe/zc1l6d4m1TSdHvtM0+SCK31BDHcn7JE0jocEp5pUuFyoOAwGRmsqr9toWr3mlXGp2el3txp9scT3cVu7RRHg4ZwML1HU96WlmGpbuPFur3Xhi38P3D2j6dbZ8hDYQeZFltx2y7PMGSOcNz0PFYtFFMC3pmpT6Tei6tY7WSQAqFurSK5Tn/YkVl/HFbWqfEDXtasxa6l/ZU0awC2jI0WzR4ohnCo6xBkAycbSMZ4rn7a1uL25jtrOCS4nkO1IokLM59AByav6r4X1/QoI5tc0PUtNikbakl5aSQqx64BYDJoe2oLfQy6fDK0EySoFLRsGAdA6kg55Uggj2IwaZRQB1cvxK8RT2dvazroslta7vIhfw/YFItxy21TDhcnk461Q0Pxhq/hu/e90c2MN00olEz6bbStE3PMZeM+X16LgVX0nwxr+vRSSaFoepakkZ2u1naSTBT6EqDiqd9YXml30lnqdpPZ3URAkguIzG6ZGeVOCOCDRsw3Rf1fxNf65AsV9DpiBX8zdaaVbWrk4I5aKNWI56E4/KrOneN9d0zTIdOhntbi0t3LwRX1hb3YhJ5OzzkbZzz8uOa5+ijYHrqzpbX4heJrTWL3VFv4p72+i8mea7s4bgmPBBQCRGCqQcbVwCOMYFV08Z65HYadZpcwLDpd2byyxZw7oJS28lW2bgpbB2528DjgVhUUbAWNQvrjVNSub++cSXN1M80zhAoZ2JZjgAAck8AAVFBM1vcRzRhC8bh1EkaupIOeVYEMPYgg96ZRRsB1lx8S/Ed3LFLdjRp5IY1iieTQLFjGijCqCYeABwAOlYWt65qXiPVpdS1u7e7vJcbpHwOAMAADAAA6AAAVQooAK19C8U6v4b+1LpFykcV5GI7mCaCOeKZQcgNHIrKcdsjjJrIooA2te8Ya74lW2TWL7zIbXP2eCKJIIos9SscaqoPA5AzWiPiZ4qDiX7fb/AGsR+UL46fbm724x/wAfHl+bnHfdn3rlKKPIBXdpJGeRizscszHJJ9aSiigDd0zxnrek6aun29xbz2cbmSO3vrKG7jiY9SizIwQnvtxmquveI9X8T6h9t12+kvJwoRCwCrGo6KqgBVHsABWZRRuGwUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB3vxO/5hf/bX/wBkrgq734nf8wv/ALa/+yVwVdmO/wB4l8vyOTBfwI/P8z3LQZT4o+CWi+EdRleafUpL4aZJK5YxXEHltFGCeispkTA/vCjWLZ9G+COp+CbJSLy1m02TUApwZLq4Z3Mbeu0LCv1X2rhLrxFoLfDzQ9MsdR1i31nR5pruORLKNYzLIUOBIJ9y7SnDbcn0FLbeLNNtfAus276lq954h1W6tbxpZ7VPLSSF3JzKZiz5D5yUHIxjnI5XZ3/7d/Bq/wCF/uR0xuoJdfet827fp97Garpng3wzq8miatDrWpXto/k3t3ZXkUESSjh1jjaJi4U5GSy7iOgFaejeBvDy+Oo/Dmty390mp2yXOj3trdR2yzCSMvErq8TlSxwmQeD2NZuqap4K8Tau2t6xJrenXt25mv7SytIpopJTyzRyPKpQMcnBVsZ7isjxF4qm1jxJBqdjEbCOxjhgsIlfcbeOEARjd3bjJPqTUq+l/n6eXne1vncp215fl/wf1/A2fDmheGrvS/Euo69p+sQ2+kANGItRiRt7vtjt2DQHLEhiXGAAp+WtPTvhqsPhPTdSvvDfirX7rU4hcxpokeyCCEkgb5TFJlzjO0AYBGetUvH3jrSfEWkw2fh6xuLEXd22p6ssoULJdsiqdmCfkHzkZx988VU/4SDw74h8NafYeLRqVpqGlQ/ZrTUNPhjnEsG4sEkjd05XJAYN0PI4o1t93/B/H8A0v9//AAPw/ETx54HPhe203Ura11SzsdSEipa6xbmK5gkjxuVuAGU7gQwAznGARXG1ra2PDiR20fhxtUmdQftFxfrHGHPGNsaFtvfq7Z9sc5NIbOo8P+HtNbw3eeJfEz3Y0uC4Wzhgsiqy3NwVLbQzAhFCjJYg9QADXV6cvhmT4NeNbnw2dSt5ililzZ38iTFP9JBDJIioGB5BBUEY75rlPD/iHTV8N3nhrxNHdnS57hbyCeyCtLa3CqV3BWIDqVOCpI6Agity38ReB9L+H3iDw7pn9uS3mrRQt/aFxaxKDJFJvWPyhKdiHGC25zznHGC3v/Xz/r7iOn9f1/XYp+CrbS9R0XXoITrVjqVvot1cS3FtqSpBcKmD5TxeVuKHjIMhzj8uX0DToNX8Rafp15ex2FvdXCRSXUv3YVJALH6fh9RXVeEdV8H6Bbag95qeuPcalpU1hLHFpMJSFpAMsGNyC4GO4XPtWRpr+D9P8W2ct9/aetaEqMbmKS2W0mZirBQoWZuA2053Dvx6u9ppvb/gseri/wCuiOyvPAXhvTLq9XXfD3jbRtPt0fytWuAksEjhTs3bIMKGbAyHfr+IyvhVp15q6eLtP0y3e5u7nQJI4YkHLsZ4cCjw74h8HeD9bl1nR7nX71/Ikjj026tYY4pN6ldssqytvQZyR5YzgdKzfBviXT/DOj+IJGudQTVdQsHsrZbaFQiZeN95l8wMD8hGAp65z2pK3vX/AJWvnZr/AC8i76prv/kQ+KLHwvo0CaZpwvrvWYUVb27W+RrRZgfnWNPJDMO2d3XpuHJ5euu8Ua14f8S6Xb6nJJfReKDHi+2WUa212+7/AFhYS7lfbyxCYY9l5NcjSV+otOhZ0/U7/Sbr7TpV7cWU+0p5ttK0bbSMEZUg4I7V6TodkbH4T+Kbm312HxALy3QTaXaM5+xnzEY3MqyhWypG3cisMty2K8+0RtFGo48SR372TIRusJEWRG7NhwQw/wBnK/Ud+ntdf8K+F9F1ZPDJ1bUdT1S1ksTNqFvHbxW8EmN+ESRy7kDGSQBnOKb+FryEviT8/wCmcPToygkUyqzICNyq20kdwDg4+uDTafCI2mQTuyRFgHZFDMq55IBIyfbI+ooE9j1bxHqHhaL4T+DIn0jXDYzvfSx28etRIQ4lClnP2Yhz6HauBxzXI+A9I0HxJ46tNF1W21EW+o3Kw27217Gj24JPLFomEnGOgT+la2q6t4I1PwjoGh/2v4gi/sb7R++/sWBvO82QP937X8uMY6nPtWZ4B1bw74c8ZWmt6vd6ns067WaCK1sI5DOoJ+8WmXYenTd9acLc3vEzvyPl3E1a18CafHeWFpca3e30MbCPUE8pbeSYHhfJZQ+zGRv3g99napdQ0Xw14US1s/EkGq6jq00EdzcR2N5HbR2qyKHRMtFIXbaQT90DOOetY2uR+HcGbQ9T1O7nklJeO802O3VVOTkMs8hJzjjA+vatq/1vwx4qjtLzxLNq2n6tDDHb3EtjaR3MV0kahFfDyxlH2gA/eBxnAzikr8uu/wDX/A/q5rO3O7ba2/T8Lm9oPw+8L32qJNf3WqvoV/pU+p2d1A8aTQCDd5sUiFGDsCuMgqD174E1t4O+HtxaeGrzf4kSPxLdtZ21uZ4C1uyS+W0jv5eGHzphQoP3vm6VW0nx94btr1raVNUstGs9IudL0+KK3juJXM4bzJpSZEAbLA7RkdsjGTnW3i/Q4NF8FwE6g1z4b1SS4lX7MmyaF5lkyG8zIcBANpGMn73HIrcyXT/gv9LEa2ff/wC1X63/AOGOP1nTm0fXb/TXkEjWdzJblwMbijFc4/CqXStLxJqEGreKtW1GzEgt7y9mniEqhXCu5YbgCQDg8gE/WotHGmnWLb+3ZLmPTw+ZzaRLJKVHOFVmUc9Mk8ZzzjBUL2XNuVK13Y9psPGF9p2m+FPCXizVr6/i8SQSS6q15cvKYorkeXbgFj8oUASY968U1XTbjRtYvNMvV23FnO8Eo9GViD/Ku78UfE/UdQ8XNceHvGHiew0aZx/o8ZNv9jjBwEjiScq21AP4kyc9OtYfxF1/SvE/ildW0hryR5rSFb2W8gWFprhV2s4VXcAMFU9epP1I9Xzf15fdtfqC0XK/67/f+Fjla9N8L3Hhxfg34m+16Vqkuy7sBdeVqcaea+ZtpTMDbAOcg7856jv5lXV+Ftf0e18M674f8Qi+ittUaCWO6sY0leKSItgGN2UMp3nPzA8Cq3hJen4NP8hLR3KVtpdp4o8VafpPhSyurM3kiwqt/eLOQ5PLFlijwoHbaTwee1aTRfD5NUOnH+3miDeUdXFxFsz08wW3lbtmedvm5x78VEdb0Hw3rWkal4HXU5bywmE8lxqgjRZSMYQRITtXrk7yTntU7TfDt9TOokeIFiJ806QIIdm7r5f2nzN2zPGfK3Y9+aFbS+3X+vv/AKsGuvyt+N/LsYHiPQrnwz4kvtGvirTWcxjZ0+647MPYjB/GsytPxHrtz4m8SX2s3wVZryYyMiD5UHZR7AYH4VmVEb8q5txu19D2u91XxzqHgHwRc6R4wlspJba4E8l14iS0aQi5cKWEkqs4CgDoeBj2rlfjLbSW/jGz+1xKb59MtzfXkMWyG9uMHfNGQAGU8DcAASppmr6r4J1fw14f0ptW1+BtHt5YjINGhYSmSVpM4+1DGN2O+cZ46VmeL/Eelajo+haH4eivP7P0aKULcXwVZpnlcO5KqSFUEcDJ781c7OTa/mb+V2C/T/I5SiiikIKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDvfid/zC/wDtr/7JXBV3vxO/5hf/AG1/9krgq7Md/vEvl+RyYL+BH5/mFFekR/D3SNT+DaeJtJkvk1uLzZJrSWZJIpYYWRZXQBFKkeYrYJPAPWmTfD3TLD4LP4ov57v+3JHhkhtUkVYo7eV2VHdSpZi3lSEYYcFT2543pe/S347HXH3kmvP8Nzzqiuktvh/4hure2lFvZwG7QPbw3epW1vNMp6FYpJFcg9iBz2zTNK8C+INZvb2zs7SGO6sAxube7vYLWSIKCWJWV1OABkkcDvQBz1FdDpPgbW9ca+GmDTpfsG83G7VrVAirjc43SDcgyPnGV96bpXgjXtZ0j+1bS2t4rDzPKW5vb2C1jd/RWldQx+maAMCitHXPD+qeG9QFlrdo1rOUEiZYMsiHoyspKsvuCRwazqACitPQ/Duq+I7mWHR7XzjDGZZneRYo4kH8TyOQqj3JFdsngSTTfhT4h1HXdEtPtFsttLp+rWt756yb51R13RStE2B2xkbvpRsK55tRXWeG/Dc1xoeq6lLpFlq8I06Z41XWoYZrIqR/pBgD+YwXB+Urg7gfSuYtbae9u4bW0hee4mcRxRRruZ2JwAB3JNO2tuo+lyKiuwHwq8XNNcQx2FrLPaoz3EEOp2sksKgEkuiyFl4HcD0pfBmn6Xqvh3xVFqGlwTXFhpT31teeZKskbiSNAuA4QrhieVJz37VN9H5K/wB2v6DSbaXfQ46itu/8I6vpmhWusXyWcVpeRrLAP7Qt2lkQnAYRBzJjOQfl4wc4rEpiCiremS6fBfLJq9pPeWwBzDBcCBmPb5ijceoxn3FdN8RdM0rT7nw/PoenLp0Oo6JBeywLM8gEjvICcuSeij246UPRXFfWxx1FFOjQyyLGpUFiFBZgo59SeB9TQPYbRXq6eBbbQfAuj3s2k+HNY1PUpZ2lk1HxJDFDCkbBVWPy7mMSE9SQzY6ECuJsvDmo+LfElxZ6BY6Zb3HmBEsotSjRC3TETTSkyZIJ+Vm6+mKOtkHS7OeorpZvh54ottKk1C40wRRRQG5eJ7mITrCDgyeTu8zZyPm24wc5xUGn+Ctb1HTItQjitLa0nYrDLf6hb2gmwcHZ5zrvAPGRkZ4oDbcwaK6fTvhz4q1TXLnRrPS86lbIJHtZLmKORkIzuQMw8wYGcpkYwe4q8vwg8buyhNIiZXIEcg1C2McrEsNiP5m13yrfKpLDHSjzA4qinSxSQTPFMjRyRsVdGGCpHBBHrTre3lu7qK2tkMk0ziONF6sxOAPzoWuwPTcjoruNU0rwR4Z1E6Lq39t6pqNrIYr+5sbmGCCKQcMsaNG7SbTkZLJkjtWR410nQtH1qCDwvqjapYyWcMxuHK58xlyykL90g/wnJHQk0rodjnqKK7jQ/hzJq/gDU9c+12KXMEtstsj6xaRpsk37/NDODG3yrgMVJ54PZ9G/63sLrY4eir2qaPc6RdpbXUlnLI6hgbO9hulxnGN0TMAeOmc1t/8ACt/E+8RfY7YXZTeLA6jbi7xjOPs/mebnH8O3PtR5gctRSujRyMkilXU4ZWGCD6UlABRXpvgzwj4K1/w5Dfa4ms6Y01/BpcNwmoRPHNcOpLNsMAKovykjceGxnvXnepafcaTqt1p16my4tJnhlX0ZSQf1FD0dv66f5oFqr/1/WjK1FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAd78Tv+YX/ANtf/ZK4KvQPilI0n9l7gox533UC/wBz0rz+uzG/7xL5fkcmD/gR+f5nsNg2v+Fvhl4L8QwaFe3VpZ3d/Ldo1u4iktpRGp3tghUdSwDHjuM4qW4v9R1rwB4o8Tan4furLR5b3SfskDRMYRaxGRAiOQAwA2gkcZbtmvGaK5eZ/l+DT/T8X3OpK0eX1/Ft/r+XY9H8ZeC/E3inxvf61oGm3WtaZqtw1xaX9qhkhETH5Vd+kZUYUq+Nu30pNf8AFsehfFLRtUsZ47+fSbO1tb+eJg6XbpEEmG7+IFSU3d8ZrzmioStZLZf8N+pT967fU9V8Y6Na/DvwnqNtp06yt4qug1lIjZxpiYkQ57F2ZQfXyzVqSNvE/gbw5ceG/B+n+JzpdgLO8tt901zayiRiW8qGdMo+7cGCnnIJ4ryCinbS39aaf8P56i63/rXX+vI7fx1JrkGhaLpmteGrHw9bWzTSWltFJKZhvKl96SzPIgyAQCFGS2M81xFFFAzvfDdrL4i+F2peHNBTzNbGpx3z2qECS8t1jZdqD+Mox3bevzZANb+jeDfE2i/BXxwmq2NzbNcR2dxHp8inzgkc+XmaL7yKFB+ZgOAT0FeR0UPdvv8A1+hNtLHpvw18H+Jvs/iC6/4R3Vvs954cuktpvsMmydnVdoQ7cMT2x1rB8MeFvFln8RNJ0+3tpdC1ssbm0bVIXgC7Az7iGXp8hHTHr3rkKKPtcy/rf/MLKzT6/wDAPZtK8Palc69dP4k8Er4Rt/ImebxHp0tzbJb/ACH5g7StC6tnaUQfMGOKxfhEsmm2Pi3XrzRW1HS7fRnSQSowglcTQt5TOBjJHO3uPavM6KFdXt2a/P8AzKvd697nefEDw7q94w8a20WoX+gaoomivJrcj7KN2zyZMDau04VSMKwxt9BwdFFJJLRBdvct6ZpOo61eiz0ewutQumBYQWsLSuQOp2qCa9F+JvhHxIuneGLpvD+qi3sfDVul1MbKTZbsrSFg524UgEZBxjNeX0U3qrf11JW9/wCugU+GGS4mSGCNpZZGCIiKSzMTgAAdSaZRQM9W8S+BvFs/wv8ABNrB4X1qS5tRf+fCmnyl4t04K7lC5XI5GetZPwg0DWbj4qaNcW+k30sNhqMYu5EtnZbYgn/WEDCdD1x0rz+inF8ruTOPNFx7nReLNB8U2WoXOqeJ9J1e2F1csPteoW0qCVzk43uOTgE/QV0vi7w5rXjO80/WvCGm3Wr6U9hbW0aWMZlNm0cSo8TouTHhgxyQAQ2QTmvOKKlK0eU0lLmm597/AIu/6HvHhvT9QMtpo1tBNf6xo3hXUbe+kswZvIeXzDDbl1z84HAXPU4HSsXT7O+Twv8ACmU21wtvBr08cz+WwSOQ3UWFY9Ax2tgHng+hryGimtJc39bt/qRZcrj/AFsl+hu+OFKfELxEjKVK6pcggjGP3rVn6NfjStesNQZDILS5jnKA43bWDY/SqVFKHuJW6Dl7179T0DxL4E13xF4rvNW8I6dc65perXT3NtdWUZkVBIxbZLj/AFTLuwQ2Omelcn4h0GXw3qv9nXV5Z3VwkatMLObzVhc9Y2YDBZe+0kDpnrWXRQkkrId7u7Cu/wDBWnXeu/DPxfpOj273upPPY3EdnCN0skaGXeyp1bG5c4yeRXAUVX2XHv8Ao0/0Fsd3oehXnw+8c+G9U8b2S2Nt9sSRoJXRpo1Uj52iBLKATnkDO002X4ceNX8SNcLZXDRtMbga4H/0Qru3ef8Aafu4/iznPtniuGooTas+39foK2/n/wAH/M6T4h6nYaz8RNb1DSCGs57pmjdVwJOxfH+0QW/Guft7ee8uora0hknnmcRxRRKWaRicBQBySScACo6KiMeWKRTd3c9c8cWVj4Q0Dw14Z8R+G9fIsbczm6t7xbSKa5m2vKF3277ynyLkNxjGPXL+MulXA1jS/E0+l3ul/wBv2gkltb5CssdxFiOQHKrnPyNnaM784Ga83oqnr99/+B/XZCWisu39P+u7CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAO9+J3/ML/wC2v/slcFXe/E7/AJhf/bX/ANkrgq7Md/vEvl+RyYL+BH5/mb1z4K1+08Iw+KJbJG0aaQRrdRXMUgDn+FlViynj+IDHTvTIvB+uzeDpvFSWONFhmEDXTzIuXJAwqFtzfeH3Qe/ocel+GNUsoPhf4b0TW50g0jXp9SsrmWQ4WBiYTFMew2SBTk9AWqxrFxZ33w81rwto15DNpej3Gk2EV1EwaOWVmmaaYEcEF2PPoorlcXrby/FpfhdfedMXeHM9N/wb/wAvwPEaK77xP4gXwr4svdB0TRNGWw0q4a1xe6XBcy3JRtrPJLIhfLEE4UqADgYxmtyxfRfDPxI0y1k07T10TxHY29w0V5Yw3Tae9xF8pR5VY4RyG54K9QalWdmuu3rv+Nv8ypXjdPpueS0V6fpou/CXh7xfqPiDTNIlnivTp1pFcaRayD7aTl2TdGdqoikhRhMuvFWzZaZ4Y8HeH0tNQ8K2d3qlkt/cXOtaXLezOWdhsT/R5Y1RQuMD5s5zjii6tf0/HX8tQd07ev4afnoeS0V2Xjx/Dt1babe6Pd6XJqsgkTUYtHtp4bbjHlyKssabWIJBVRj5c8ZrjaBhW5Y+FptQ8Ian4gttQsymlmP7TaN5gmAkcIrD5NhBJ/vZ4PHTO14ehtdD+Hd54saxtNRv31FdNto72ESxW2YjI0pjb5XY8AbgQME4Nb2neIhr3wZ8bC40jTrS8t0sQ11YWqWwmU3IwHjjATIIOCAOCc5xR5E+Z53pWk/2ql839oWNl9jtHucXk3lmfaQPLj4O6Q54Xvg1n13vgK8GoaR4h0u+sdMnt7XQby4hd9NtzMkgAIbztnmZGTj5uPwFcloE2mW/iLT5tfgkuNMjuEa6hiOGeMEbgOnb3H1FPRyUf63YX91v+uhn1uaL4Wl13RdV1C01GySTS7c3M1nL5oleIFVLqQhQ8uBgsD7Yr0qxvbXU9ZvY9Gg8D+JYpIJTbaRb6ObK5A2HBjd4BudfvbS7k7Tjnkc78IrCHVJPFlldX0OnwT6C6SXc/wByFfPhyx+npSScrpdm/wAH/kVomr9zzuiu18fahJp96/ha30az0+005VhEr6fbi7udvImeZU3HcMEbWwRjJbkniqSd9UDVtzT8P+H7/wAT6zFpmlRq00gZ2aRwiRIoyzux4VQASTWrqXgW4tNDutW0zWdJ1y1sZBHef2dLIWt8nAZlkRCUJ4DLkZ71habLqK3RttHkuhPeIbYxWrMGnViAYyF5YHj5e9dpfLbfDvw5qOieal34m1eAW+oCNw0WnQ7gxhyOGlJUbuy4x1zTlov63/y/L8Ajq9f6RwFFFOjcxSLIoUlSGAZQw49QeD9DQI6Ox8FyT+HLfXNU1rS9FsruV4rU3zTM9wU4YqkUbttB4JIAzWFfW0VpfSQW97BfRoRtuLdXCPx2Dqre3KjpXp3iXxhqUPwv8E3KW2imS4F/vD6FZMg2zgDahiKp77QM9Tk1gfCy7+1fFDS7S8stOurbU71EuYbnToJUKliSFDIRH1/g2+lOK5pWRMpcsXJ9DiKK6/WfGNq327S9M8L6Ja6aUeGJXtFkuIzu4l+0cSFxjpkJzjbitDxNfr4GvLXQdE03SpDFZwT3N7fadBdyXUksSyEgyowRBuCgLjpkkk0lqrmko8snDqr/AIafqcBRXtvhe20gXNp4gPh/SpINW8P397caZPaJLCs9sXAaMsC0asQDtUjuOgGEstasJtN8B3cvhTw21z4i1GS1v3GlRBTCk6oFRMbUJEpyygMdq88Ubu3X/gtfoRdWb/rZP9fvPE6K1PE1jDpni3V7C0UrBa300MQJzhVkIHP0FM8Ox203ijS4r/b9le8hWbecLsLjdk+mM0U/ftbqE/cT8jWtfhz4nu7G0uhYwW6XwzapeX9vbSXAPQpHI6s2e2Ac9qytd8Par4a1BbHXbN7K6aJJvJkI3BWGRkA8H2PI7itP4jS3cvxM8RG/3+cuozIA/wDCiuQgHsFAA9sVk6xrmpa/dRXOsXTXU8MCW6SOAD5aDCgkDnA7nk9zUp8yTX9IpqzaZQq7b6PfXWjXmqwQb7KykjjuJd6jY0m7YMZyc7W6DtzVKvTfC/ia+tfg14m8qDS2+yXVhHH5uk2sm4MZs790Z8w8cF8kc4Iyat/A5drfi0v1Bas8yorqdAifx98QNG07VTa2sd1OkDtY2UNqNmcnCxIqljyASCenpViTx0qau0S+GdBGkK/ljTzpsJfy+mDcbfO34/j35zz7UJN2XcV9/L9f+GZx1Fb/AI60K38M+OtW0eydnt7W4KxF/vBSAQD7gHB+lYFRGSkk11G1Z2Orb4a+Ikt7W4kOjxxXilrZ316xUTAHaduZucEEcd+Kw9b0PUvDmrS6Zrdo9peRY3Rvg8EZBBGQQR0IJBrvtXt/Ds3w48BHxFqWp2hFpdbUsdPjnDL9rkzlmmTafwP9KpfFaOG4k8P6lo8q3Hh99MSy0ydifOZYDtcSggYcMx6cYK4Jq5K0muza+6/+Qf5Hn9FFFIQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHe/E7/mF/8AbX/2SuCrvfid/wAwv/tr/wCyVwVdmO/3iXy/I5MF/Aj8/wAzpbrxn9s8G2fhyXQNK8my3tBdK1z5yO+3e/8ArthJ2jgrtHYCmReL2t/Bd54bttG02GK9aF7i8UzmeRomJVuZSgPzMOEAwemcEc7RXGdex1reP3unguNa8OaHrF/AoX7dexTebLgYUyBJVSQgADLqScc5rn9Y1i+17Vp9S1SczXU5Bd8BQMDAAA4AAAAA4AFUqKOtwOm8W+PtX8Z2unwaslsi2KEbrdCpncqqtLJknLkIuSMDjpTdN8a3FpoC6JqelabrenxMXt4tRjk3WxY5by5I3R1BPJG7B9K5uigDW1vX11iO2ih0fS9KgtgQkdhAVLZxy0jszv0/iY47dTWTRRQBteH/ABRd+H47u3W2tL+wvVC3Njexl4Zdv3W+UhlYZOGUgjJ5rbf4mXP/AAiuoeHLXw7oVnpV/GqyQW8UwYOrbll8wyl2cHGN7MvGMYJB4qigVjrND8dQ6BazRWfhPQ5JLizayuZ5XvC88bAB8gXAUE46qBjtiq2m+MX0TxbZ6/oWi6Zp01ojKlvH58kLllZSzCWVmJw3Tdjgcdc85RQ9XcfSx2EHxDksLuW/0bw1oOl6m6Mi31rFNvi3DBKI0rRocEjIQYzxisvRfFMuhaLqthaadZPJqlubaa8l80ypESrFFAcIOUByVJ98Vh0Urb/cB0Oo+L31XwvY6NfaRp8klhF5NvqRM5uUTdu2583YR1ABXABOADzXPUUU+tw6WN/wf4uuvBmqz6hYWVndTTWz2/8ApQkBjDYyyNG6MjYGNwIOCalm8W2T2V5BbeDtAtZLuIxNcr9qlkjyQdyebO4VuPvYz19TXN0UPVWYLQKfC6xzI7xLMqsCY3JCuM9Dgg4PsQfemUUAdld/EK3vtI0/TLrwZ4fez03zPssXmXw8vzG3PyLnJyeeScdqz/DHi4eFdaXVbTQdMuruKcT2z3TXBFsRnAUJKoI/393SudooWjuhNJqzNjWdbs9ViAtvDmmaXL5m9prOS5Zn6/KRLM645zwAeOtXoPGztptpZ61oWk64LJRHbTX6zCWOMdI98UiFlHOA2cdBxxXM0UbKxTbbuzs7P4m6lbazdajcaXpd401k2nxQSxyxw2tswIaKJIpEwCGPXJ75zkmpF47u4dM0Ozj0zTx/YN817ZTYl3qWkDmM/vMFCVXtuwPvdc8vRRs7/wBdxdLf1tb8i3q2pS6xrV9qdxHHHLe3Elw6RAhFZ2LELkk4yeMkn3qpRRRsB1aePZJltn1zw/o2uXVqqpHd38c3msqjChzHIgkwABlwxxxmsjxD4i1DxPqv2/VHjLrGsUUUMYjjgjX7saIOFUdh/WsuijcNgrd8PeLLjw/ZahYGwsdT0/UlQXNnfI5RihJRwUZWVhubBDDqawqKAN/V/F1xqUlibHTNM0RbB/MgGl25iYPx85kYtIx4HVjjFaP/AAsRzqH9pt4Y8Ptq/wB46gbaXeZP+evleZ5O/POfL684rj6KAJru7uL+9mu72Vp7ieRpJZXOWdickn3JqGiijYNzsbrx/b32l6dp954N0Ca202No7VTJfAorOXbkXIzliTz6+lZXiXxZeeJRZQy2tnYWWnxGK0sbGMpDCCcsQGJYsx5JJJNYdFD1d36gFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAegfFIRj+y/LZm/12dy4/ue5rz+u9+J3/ADC/+2v/ALJXBV2Y3/eJfL8jkwf8CPz/ADPZdK0bTPE/wT0/Tm02xTX7j7ZLYXkNpHHNM1tsbyWZVBfdGz/eJ5UUzUvD+maJ8DbvTV0y1k8Qo1hc3d01urTxG5Z2WFWI3JhI48gEfePqawm1qy0/4d+GG0XxVZQa7ol1cXwgWC48wNIYyqqTD5ZYbDkFtvuafH4qibwT4g1DVPEtpfeI9WvrK+W2EEwfdC77gx8pYxgMpAVsYBA7A8rs7/8Abv5q/wCG/p5nTG6gu/vW+92/4Hr5GPeeFPD2h3w0zxH4lurbVUwt1FZ6YLiG1c9UeQzKSy/xbVYA5AzirmjfDi2u/F8vhvWNZmtb14PtFg9lZLcxX0flGRSjNLHgso+UEYJ4JBpNctvC/ivxDPr6eK7fSF1KVrm6sby0uJJreRjl1Qxxski5JwSy8Yzis/xF4vFz4w07U/D/AJsEGiw21vp7zAeYVgA2u4HGSQTgdM4qV0v/AMN/wzt1KfW39f8AD9STw54U0LW7LXby71zUbG10iMzGX+y0kEkZYKgP78bZGY4Ccjg/NTYfCWkWHh+y1LxXr0+nTainnWllZ6f9qlaHcVErbpI1UEg4GSSBmtnx74l8NSaC1l4LlZhrd8dU1NDEyG3baNltkgblVmlPGRyMGrdn46l1XwdpVlb+O9Q8I6jpNuLTyhLdC1u4wxKOPIDFHAODlTnA5o1t93/B/HT8fQe/3/8AA/DX8PXjfEvhmPRYLLUNM1FNV0jUA/2a8WIxNuQgPG8ZJKMMjjJBBBBNYFdT4y1Oa+hsY7nxxfeK5o9xfzvPMNuTj7jTEMScc/IvQde3LUhs6Hw/4Yh1PS7zWNZ1H+ytHs3WJ7kQGZ5ZWBKxRpldzYBJywAHU9K7qxt7GD4F+MRofia41XT/APQibG5ga3e1lNwuW8ve6fMAPmVj93B6CuT8P6tpd/4LvPCeu339lxterqFnfGFpEWUIUZJFQFtpXGCoJBHQ5rcsW8IaL8M/FGjw+LIL3WNUht3QpZ3CW/7qXf5Ss0e4ucdWVV5HPUhv/L+v68vnH9f1/X/AzfBFrFLoniA6P4gvbLUv7EuWvbZtLieGa3UgtGJTLuBbC8hARzz68roOjz+IPEFhpFo8cc99cJBG8rbVUscZJ9Oa7HwRHoOkW2qzaj4v0mF9T0WezSD7Pes8MkgGA+LcrgY5KlvbNYemaT4ct/F1la674jjn0d1Z7i+0qKbdEdrbQBLErE7gvRSMHqOzv76vt/wWPXldv60RsL4L8IvrN3pTeNbi0u7VJCWv9H+zxSsiltqsZsgkjA3quc+uAZfhrdajceHvGmkW091Lby6JJIllG7FHmM0Khgg4L4OM4z2rZ0vxTa6deTP4k+I0nirQvIdDo88V3I9zlcIpWZPLjIODuDkjHGawPhz4jg8I2HiLVU1uKx1SbTXtLGBYpWmaUyRuGBCFAuFYcsDnt3pK1pKW3K/yf/AL6prv/kZfiTw3pHhy0itpNZupte8tWutPFinlWrn70bTCY5YegX2O05A5iuw8Vjw9rOnw+IdN1Oxs9VuU3aho0cM4xNuwXiby9gBHzFS2F5AJ4A4+kr9RadC3pmp3GkXy3dkIDMgO0z20c6j32yKwz6HGR2rtPixeXGoz+E7y9laa4n8N20kkjdWYvLk1xemWcF/eiC61K102Mgk3F0srIPbEaO3P0rt/HQ0HV9P0STTfF2lTyaTokNlJALe8V5pY2cnYWgC4O4YLFffFOWy9f0ZK+L5f5HntOjCGRRKzKhI3Mq7iB3IGRn6ZFNp8KLJMiPKsKswBkcEqgJ6nAJwPYE+1APY9jn1LSPC/wu8MLoPizxHo0OoSXc0txp2mRxzXbJIEHmbbpSAo4ADMD14NcN4c0u28feO1sNZ8QamLrUbhYbe9ltBcvMScAy7pgV4A6F/TtW3rcHh3UvA3hjRoPG+irc6R9r89ntr8I3myh12kW2TwOcgfjWX8Nn0jR/Hun6xrHiCxsrXTL1JDvhuXa4UE8xhIm/8AHtp5pws5LmJndQfLuQap4X8OaTDd203i9JtXto2Yw29i0ls7g/6oThslyO+zZkY3d6bL4V0fRrO0PizW7uxvruFbhbKx09bloYnGUMhaWMKWByFG44IzjOKzte0uwtC1zY+I9M1UyzH9zaRXKugOTuPmwouO3BJ56V0GtS6D43ktdYuPEVromo/Z4be9tb23ndWMSCMSRNFG+QyqDtbbg55IpK7jd7ms7KbS21/4H4XLWjfC+z1LWzaXXiQW9hcadJqVjqcdkZIZoY93mbwXVo2XaQVAY5z7E2IPhj4buLfRrqPxvILbXbg2unbtIYSPKH2NvXzcIgJT5gxJ3fd4NaGieJ/DMEy6Smux2umaXod5p1vd3ltNuvJ7neWcJGj7UDEcNg455OQMu017Q4NB8AK+sQedoOryyXsIhm3CJ7hHEinZtK7UJxndyOOuBayS6f8ABf6WI1s31/8AtU/zv+RwGpWE2lard6fdbfOtJ3gk2nI3KxU4/EVWrW8V3dvqHjLWryxlE1tcX88sMoUqHRpGKtggEZBBwQDVbR7WzvdYtrfU9Qj020d/311KjuI16k7UVmJ7DA6kZwMkKF2lfcqVk3Y9g0m40yfRvDWgeKdK0X7f4rjnJu4tItoJbKNx5dqymONeTIpbPXBrxi8tJrC+ns7tDHPbyNFKh6qynBH5ivSfF/j15fG0f/CP6n4bl0pTHFZXD6FHKbKFPlQO01t5pKgBjt3YzxnpXP8AxPv9I1bxo+q6JfwX32+3invWtoZI40uiMShRIiMQWG7OP4/rQ9XzLr/S/wCCC0XK/wCu/wDwDkK9G8N6R4SuPhP4gvNQvL5bmK5shJOmkxSPbFjL8sTGcFlbHJOzoOD285rsvCWoaRJ4M8R+HdX1RNJl1B7ae2upoJJIi0JfKN5aswzv4IU9DVfYkvT81+gluYjaRa6lr1npnhKe91J7tlijF1aJbOZGOAoVZZBjpyWHfjjNbDeGfCial/ZT+MJftwbyjcDTM2Ik6Y87zd+zPG/ysd8Y5p1rd6J4F8S6Jq+g61/wkF1aXAmuES0eCEKMfIrSYckjdzsAHHWlbQfBj6obweM0XS2Pm/ZTYz/bQOvl42eVu7bvMx39qFbS/wA/6/4cNdflb8f+Aczq+lXeh6xd6XqUXlXVpK0UqZzhgex7j0PcVTrZ8X+ID4p8X6lrRh8gXkxdIs52KOFBPc4AyfWsaojdxV9xu19D2Oe38T3/AIJ8Iah4V8H2GoS3tvOb2a38LWsys63Dou4iEhflAHb1965L4rabaaX4vggt7G3069awgk1OxtRiK2u2XLogyQBjacAkAkitDXIfD+q+EvC2nW/jbRorjSLaaOcyW98AWedpBtItjnAYA9OfzrP8f67p2oaT4d0qx1WTXLjSreVLjVZI3Tzt77ljXzAHKoMgFgOpwBVztzO38z+67/D/AIAL9P8AI4miiikIKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDvfid/zC/+2v8A7JXBV3vxO/5hf/bX/wBkrgq7Md/vEvl+RyYL+BH5/mFFd03w3Sf4UJ4z0zVXuJEci40+S1CNGqsqu6sHbcqs6fwjhs8Uz/hXJh+Eb+NbzU/JkaZEt9OW33GSNnKCRn3DaCUkx8pztHrxxvS9+n67HWtVdf1Y4iitmw8H+JtU08X2meHdWvLQ5xcW9jJJGcdfmCkVFpXhjX9dEx0TQ9S1IQELKbO0km8snoG2g46HrQBl0VtWXgzxRqTXC6d4b1e7NtKYZxBYSv5Ug6o2F+Vh6HmodJ8Ma/r8ckmhaHqWppEdsjWdpJMEPoSoOKAMuip76wu9MvJLTUrWe0uYjiSG4jMbofdTyKgoAKKs6fpl/q94tnpVlcX1y+SsFtE0jt9FUE12EfgX7L4C8Q32v6TrOlazpS28sIuv3UcySzCP/VNGG455D46cccgrnDUV0uieFbi40PUNX1TQvEMunLZSvZ3thZloBMpwDK7DHljDbiDkEfWuaALMAoJJOAB3ptWdh9LhRW9L4F8XQQCabwtrUcRBYO+nShSAM5zt9BmrnhTQtJ1zw/4je8S9TUNL0976CaK4QRMA8abGjMZJ++TkOOmMd6XR+Sv9w1q0u5ytFalx4Y1+00dNXutD1KHTZArJeyWkiwsG+6Q5G057c81l0CCigAswCjJPAA716J4n+HumeG/h2L6S6un8Q213Bb6hBuXyIGljeQRgbc71VUyd2MkjHFD0VwWsuX+tNTzuiinRxvLIscSM7uQqqoyWJ6ACgNhtFeoW3wputO8I6fqWs+EfF2qalfyy/wChadCbf7JGh2gyFoZG3MeQNoGO9cUvhzUta1i9t/DHh3WJVtmxJaCNrqa37ESFI153A/wj07UdbB0uYlFaZ8Na6NHbVzouojTVOGvfsj+SOccvjb1469adpXhXxDrtu8+iaDqepQo21pLOzkmVT6EqDg0AZVFaln4Y17UZ7qDT9E1G6lszi5jgtJHaDr98AfL0PX0NWf8AhBvFouZbc+Fta8+GMSyRf2fLuRDnDEbcgHB59jQBhUUUUAFFekad8MNJ1zwuup6P4jvTd3IuRYWN5pSQteNBHvcKyzvxjgHHUYxXm9GzsG6uFFFdlo/ww8R614NvdetNJ1STyZIBawRafI/2xJN+50YdQu0ZwD97qO50v/XYOtjjaKu6romq6FdLba5pl5ps7JvWK8t2hYrnGQGAOMg8+1XB4N8TnS/7THhzVjYbPM+1ixl8rZ/e37cY980dLh1sY1FFFABRXXP4R0fR9NtZfFviCewvruJJ49PsdP8AtUscTjKtIWkjVSRztBJwQSBmofGHhC28M2Wj3lhrUOr22rQSTxSxQmMKquVAIJzu45BAwcjnGaHpv6AtVc5eiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAO9+J3/ML/AO2v/slcFXoHxSiki/svzI2TPnY3LjP3K8/rsxv+8S+X5HJg/wCBH5/mesaN4ji8MfD3wZeXsL3OnTXmp2t/bpjM1vIIldRnjODkZ7gVYu/EWn+JPB3imeGKa30K0u9Hs7WCTG+O1j80YOCfmPzMcE8sea82fxd4kk0caTJ4g1VtNEYiFk17IYdg6Ls3bcDHTFMfxPr8uhjRpNc1J9KACixa7kMGAdwHl528EA9OvNcvN+n4NP8AT8WdSVo8vr+Lb/X8EdL8Ro9bb4mz+QlyUaYf2J9lB2m2/wCWHkbeNu3bjb3z3zWzrviCLwz8XtN1C9RZZzYW8HiKEDIlkkhCXKsBwWwef9oetcNYeMPE2l6eLHTPEWrWdoM4t7e+kjjGevyhgKx2YsxZiSScknvUJWsu34+Xz69ynrd7X/r/AIbseoa14fb4ZeF/EAeQNd63dHTdPk7vYrtkklB9HDRL/wB9VY1ubSh8MfCsw0rXL3Ro7PZcPpuqJbwRXgkbzPNQ28mHOVIYtyMYHFeY3uq6jqUdtHqN/dXaWsQht1nmZxCg6IgJ+VfYcVNpPiHWtAaRtC1e/wBMaUASGzuXhL46Z2kZp9Len4f1f1F1v6/j/VvQ6PxtrTaz4d0HOg6vaRW4lS31LVLr7Q91GSpEYkEMYZUJOOuN+OBXF1c1PWdT1u5FxrOo3eoTgYEt3O0rAemWJNU6B9EjvfD/ANo/4U9rn/CO+b/an9oRf2j9nz5n2Dy2x052eZ97H+znjFaugf8ACQf8M+eMP7S+1f2PusvsPn52bvtC7/Lz2+7nHGcV5tp+p3+kXi3mlXtxY3KZCz20rRuv0ZSDWjP408UXRuDdeJNXmNzCbefzL+VvNiOco2W+ZeTweOTQ935/oTbQ2vht9/xV/wBi1e/yWsnwNHq7+OdJbw3ZR3+qQ3AntraTG2Ro/nwckDop7g+nNLa+PfGFlaxWtn4r1y3t4UCRQxajMqIoGAAA2AB6Cqlz4o1+91SLU7zXNSuL+FDHHdy3cjSopyCocnIHzHjPc+tO7UuZf1u/1CycWn1/4B6T4buhrfjzUbbR4fEnhrX7uCc3E91frdwp8h3ieN4VYJjI3MzFSQaxvhFJpsMniyTXIJrjT00F2uIYH2vIgnhyoPbNcrdeM/FF9p7WF74k1e5s2Xa1tNfSvGR6FS2MVUttd1ez0q40yz1S9t9PuTme0iuHWKU8DLIDhug6jtSWl7dmvz/z2Kvdpvvc6D4ixXjaxBeieG50S5izpElqmyBYAf8AVqn8DKeGU87sk5zk8hWm/ibXZNDGjSa1qLaUAALFrtzAADuH7vO3rz0681mUkraBe50ngLVtG0LxZDqniCO4kitUaS2EECzbbgD92zIzoGVT82NwyQK7C8Oh3XwV126h1rVr2WfXoZnlutNjiaScxSnBxO3ynJJbkg/wnOR5XWwfGHiVtH/slvEWrHTfL8n7Gb6Tydn93Zu27fbGKctYtf1vf9P6sJaST/rZr9f6uY9FFPhmkt5kmgkaKWNg6OjEMrA5BBHQigDtvFf/ACSPwD9NR/8ASgVV+E3/ACVzwz/2EI/51U/4WP43/wChy8Qf+DSf/wCKqjp3ivxFo8lw+ka9qdi90/mTta3kkRlbn5m2kbjyeT604PlkmTOPNFx7ljXPFevalrl9dXWq3XmTK9s4jlKL5JODEFXACcD5QMcdK3fiQl01xoh01ZT4eOnW40rygfK3eWPNAxx5nm7938Wevaua1bxX4i162S31zXtT1KBH3pFeXkkyq2CNwDEgHBIz70mleKvEOhW7waJrup6bC7bmjs7ySFWPqQpGTUpWjb+uv+ZpOXNNyXW/42/yPYdGub6w1DSrm/kmg8UQeEdRe7kckToAHNuX7hwmOvzY25rN03XNVHh34Tx/2ldbJtbnaUec3zkXMYGeecB2/wC+j615hYeKvEOlXVzdaZrup2dxdnNxLb3kkbzHnlyCC3U9fWmDxJrgs4rQazqAtobj7TFD9qfZHNknzAucB8kncOcmmn73N/W7f6/gZ8vutf18KX6fiWPGiLH4919EUKq6ncgADgDzWrJt4Jbq5it7dGkmmcJGijJZicAD8aLm5nvLqW6vJpJ7iZzJLLK5Z5GJyWYnkkk5JNS6fqN7pN9He6VeXFldxZ8u4tpWjkTIIOGUgjIJH0NKCUUk+hcm22z1jXdX8M+F/GOgWceuatDN4PSO3MNppUUsMk6tvnO83CE7nLKflHAxzXDfEXRLfRPG94mm86Zehb6wbGA1vMN6Y+mdv/Aayr3xLrupanb6jqOtajd31tjyLqe6d5YsHcNrk5XB5GD1qHVdb1XXrpbrXNTvNSuEQRrLeXDTOqgkhQWJOMknHuae+r31/Hf+ugaLRbafhsUq7nw1DJP8HPGiwRvIUutPkcIpO1AZsscdAMjmuGq7pes6pod0bnRNSvNOuGXYZbSdomK+mVIOKe8XHvb8Gn+gr2Z0Pg7RDp/jHw5feLdOmttEub5F868hZIZQCD1IwVyRntjNE0fjb/hZ7fJqP/CT/aty4DeZuzwR22Y/4Dt9q57VNZ1TW7kXGtald6jOBtEt3O0rAemWJNXB4y8TjS/7MHiPVhYbPL+yC+l8rZ/d2bsY9sUKVmn2/wCB/kK2672/X/MufEb+zf8AhZGu/wBieV9i+1vs8n7mf4tv+zu3YxxjpXNUUVEY8sUuxTd3c7r4pW8uo+Jk8TWSPPpOsQQNa3CqSu5YkRoiezqykFevT1rmdY8N6voEFlLrVjJZfbozLBHMQsjIDjcUzuUHsWAz2zSaT4l13QFkXQta1HTBKcyCzu3h3/XaRmqd7fXepXkl3qN1Nd3MpzJNPIXdz7seTVeglsQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB3vxO/5hf/AG1/9krgq734nf8AML/7a/8AslcFXZjv94l8vyOTBfwI/P8AM0Lnw/rNlpMGqXmkX9vp9wQIbuW2dYZcgkbXIwcgHoe1Mi0XVZ9Hm1eHTLyTTbdwk16luxhjYkAKz42g/MvBPcetev8AhSWDUPhJoPhK/ZEt/EVxqFvDI5wIrpDE0DZ/3sr9HqfX4Ps3wt1XwLYlCdFuNMt52B4e9maVpjn0DFU/4BXI01e3l+LS/C/4rudUXePM9N7/ACv+dvwZ4ZRXc67ceGvCniG40CLwta6uunSm2ub68urlJriRTh2QRyKiLkHaCrHGCc1s6VoHhfTPiHZaLqWmQX+l+ILSK5024vpp1ktWmjJiV/KkQMPMIVvUcgilo9uo3pv/AF/wx5bRXoehWOl2Gh+LNU8T+FNPkXTpvs9tDJNdIVu3bAgBWYZRFV2OctwPm5q/B4QsNG8J6NM+jaHrGo6parfSS6v4gjslgjZiFRIvPic8LkucjJwOho6X9Px/4GoPR2/rQ8torsvHmhaRp9tpuo6QbG0lvBIl1pVpqsV+LZkxhlkR2OxwcgMcghuSMVxtA7BWrb+GtUu/Dd5r1tFDJp9kyrcsLqLzItzBVJi3b8EkAHbjr6Gtrw/pWl2Hgu78Wa7Y/wBqRrerp9nYmZokaUoXZ5GQhtoXGApBJPUYrpNO1LQ9S+DXjWTS9DXRr6NLFbhLe4kkgmX7SMMokZmUjkEbiDx0oJueeaVoWpa2l8+l23nrp9o95cneq+XCpAZvmIzjI4GT7Vn123guDSNa0fXLG/0Cye4sdGuryLUFmuFm8xMFcgS+WQM9NnYVzOgQ6bceItPh164kttMkuEW6mjXLJHkbiPw9j9DTteSiv61YX0bM+tXTvDOqato99qenxwS2+noZLkfa4lkRBgF/KLByuWAyFIycV6SNE0D7dqBfwRpN/pcEEjxS6D4iN3dIAh2yPH9oJZQ2C3yJgZPbBxfhFpNxrsnizS7JolnutBeNGmcIikzw8sx6AdaSvK6W9m/wf+RVrNX2vY87orrfGMmh6XI3h7S9AgS7sAILrVZTcpNPMpw7CJpCiKemCue/y5wOSpJp6oGmty3pmlahrV8tnpFlPe3LgkRQRl2wOpwOgHc9BVrxB4a1XwveQ2uuWy2808C3EQWZJQ0ZJAOUYjqp468VlV3HxL/1Pgz/ALFi0/8AQ5ab2v8A11JW9v66HD0UU6NgkisyLIFIJRs4b2OCD+RoGbmmeC9b1bR/7Vt4LaDTzKYVub6+gtI3cDJVTM67iPbNZN9ZS6dfSWtw0DyRkBmt7hJ0PGeHQlW69ia9O8S+INNj+F/gmZ/CGiyxyi/2QPNe7IcTgHaRcBjnqdxb2wOK5r4cQ6TrPxFsNL1fQ7O8s9Uu1iMby3CfZ1JP+rKSg+3zlulEfedkTKSjFyfQ46iuu1jVfB8C3umaV4WeRER44NTuL6Rbnzc8SFATFs6jZtJx/HnmrWtw6H4IltdHn8PWmt6j9miuL25vp7hVVpUWQRxrDImAqsAWbJJz0FC1Vy5Jxk4vdfocPRXr/hvwv4SnvoNVu9DN3ouqaLd6glpJdSrNaS224PGjqwBUkDBcMcY7g5daWXga4sPB96/gqMSeJr57OSFdSuPLtkSYRl0+fcXIkX7xK/L05o3dv63a/NE3Vm/62T/Jnj1PggluriOC2ieaaVwkccalmdicAADkkntV3xBpyaR4l1PTYXZ47O7lt1ZurBHKgn8qz6UWmkymrOx0h+HPjcAk+DtfAHUnS5v/AImucdHikaORWR1JVlYYII7EV634m0ewuvEXhW6ufGth4flGhadgvHdGeHEK4cFI9n/j498Vy/xcEsnxN1O8kthBDe7J7d1dXW4iKALMGUlTv2luD1JHUU5Llk15tfcw6fc/vOLqaOyupbOa7itpntoGVZZljJSMtnaGboCcHGeuDUNem+F9ZsYPg14m83w1pdx9nurBZPNkuh9pJM2GfbMMEY42bRycg8U/suXa34tL9RLV2PMqK6TSrGLxx410vStP06y0NbyVIGFq0zoozkufNkdicdgQOB9auPrvg9NUNoPB0TaUr+X9qN7cfbivTzM+Z5W/vjy8dveizdg7nH0Vs+L/AA+fC3i/UtFM3nizmKJLjG9TypI7HBGR61jVMWpK6G1Z2NTSfC+v69C8uh6HqWpRxttd7O0kmVT6EqDg1VvdL1DTPL/tKxubTzN2zz4Wj3bWKtjI5wQQfQjFdl8VLu5sPFA8NWs8sOkaTbW8draK5CAmFHaQr0LszMS3U5rA8ReMdX8VWWlQa3MLhtLgaCKdsmSRS2752JO4jpnjgDOetN+QlqjCooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDvfid/wAwv/tr/wCyVwVd78Tv+YX/ANtf/ZK4KuzHf7xL5fkcmC/gR+f5nWX3irRb3wBpvh46Lfpcac0ssV2NSQo0ku3eWj8jO35OFDAjPJNNtPFem6f4B1DQbLS71L7UJbeaW+fUFKI8LMQViEQIBDsOXPODnjB5WiuS71/rY6rWVv61Oyu/F3h3W75dU8SeF7i61VvmupbPUxbw3b/33j8piGPU7GXJyeM1ha/4ivPEGuHU7gR27KqRwQ24KpbxoAqIg7BQB/Osqil10Gdn44+Ib+MrCyto9Mj00JI11emOXcLy6ZVVpiNo28JwOcbjzzUFr4v0678N2ujeLNEk1RbAFLG7tb37NcQIW3GMsUdXXJJAK5GTg1ydFH9f1/W2gGtreoaLeR20WhaG2mJCD5ks141xLOTj7xwqDGP4UHU5zWTRRQB0Ph/xPDpml3mj6zp39q6PeOsr2wnMLxSqCFljfDbWwSDlSCOCOlbiePPD1p4J1jwzpXhOe2t9Uij33b6mJLgyxvvRmbygpQHHyKq55+bJBHBUUbisdn4a8V+GfD1rdj/hH9Wubi+0+SwuZP7YjRCsgAZkX7MSp44yWx71n6brnh/R/Ftnqdp4dlvNPgRhLp2qXiXAmYqy5LCFAAMg42k5Xr6c5RQ9XceysdvpfjTw74d1c614c8L3VrqiI4tjcat51vbsyldwj8pXbAJwGkI9c1l+HfE1noGha5a/2dPcX2rWbWQuPtYSOGMsjZ8vyyWbKdd4GD071zlFHfzVh7O51OveKtN8RaBZJqGlXba/awiFtWN8uLhQ3HmReVliF+UHfnpktgCuWooo63F0sW9Mk06K9Daxa3V1a4OY7W5WByex3tG4/DbXUeJ/FvhvxHY2Ma+H9VtrjTtOTT7WQ6xG6BULFWdfswLHLHOGXPtXGUUPVWDZ3CnwmNZkM6M8QYF1RgrMueQCQcH3wfoaZRQB3WoeMPCWpeHdI0afw1rS22ked5DJrkQdvNfe24m0weRxgD8azfBviTRvCviaHWrjR76+ms7lZ7ONNRSFVAJ+WTMLF+3I2/SuXooWjuhNJqzNjW73w/eJv0XStTsrhpS7vd6lHcIVOcgKsEZBzjnJ+netV/FmjaxZ2Y8WaDdX99ZxJAt7ZaiLVpokGEWUNFIGIGBuG04AznGa5KihaKxTbk+Z7noGnfEuyttUmlutAk+wJpkul2FhZ34hW1hlDCQlmjcu53Z3cc9QRgCjB45s4NJ8L2q6ROZvDmovdwym9G2ZHlWQxsvl5DfIo3A46/LzxxtFHW/9b3/MXS39bW/Iv67qQ1nxFqWqJCYFvbuW4ERfeYw7ltu7AzjOM4GfQVTgMS3EZuUeSEODIkbhGZc8gMQQDjvg49DTKKErKyB6nda54v8ACPiC7sp77wxrINnZwWSLHrsQDRxKFUnNqTkgc4x7YrD8X+Jz4p1WCeKxj0+zs7WOzs7SNy/kwpnapc8sckkk9c1g0UPV3+fzAK6Tw14otNI0XVtG1jSn1LTdU8pnWG6+zyxSRlijq+1x/G2QVIPFc3RR0sB0t54n0+1v9NuvB2if2FPp8omW5e8a5mkcEEFmIVMDHQIOpzmrh8UeEn1L+1X8Fub4nzDANTIsjL13eT5W/bnnZ5mO3SuOoo2Auavqt3rmsXeqalL5t1dytLK+MZYnsOw9B2FU6KKSSSsht3dzq/8AhLNL1XT7SDxdolxqNzZRrBDe2V+LWV4l4VJN0ciuFHAIAOABk4rP8S67ZazLbR6RoVpotjaoUihhPmSvk5LSykbpGz0zgAcACsSim9RLQKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA734nf8AML/7a/8AslcFXoHxSdX/ALL2xqmPO+6Tz9z1Jrz+uzG/7xL5fkcmD/gR+f5nq9n4O0HXvgrBfWWmpa+J8XM8csU0p+1x2xTzFKM5XcVkLfKB9ztUd74M0PSPgbLqF1Zeb4mc2t007SyD7NBcO/lqEDBclYiTuBOJPpggv7zw/wDDnwhquj6ho7aho95d3ktq2p27SCOXywqtF5m9twDAoBuAzkCpW8QXV94J8TeItbudHF5qN/pt1BYQX8O8xws6lFhDmRQoKDBGcc9ATXK0ne2nw/mr/h+T7nTG6gm9X735u39eaOSfwG1m8NvrfiPRdHv5VVvsN485ljDDKiQxxMkZIIOGYEZ5xUmk/DjUdS8Q3uhXeo6fpWp2aNKba8MzNMioZC0ZijcONo3AA5I6A1o+JvDieK/Fl5r2i69opsNWuHus32qQ20trvO5kkidg5KkkZQMCBkdar6/4xjsvH2jan4cnNwmgWtpaRXO0oLryVAZsHkK3Iwf4aldL9fw8/l+N9Cn1trb8f6/Az9B8Er4gj1WS38SaRbx6WjzTSXC3QVoVIHmqVhPBLABThyf4abY+CZZ9Bg1fVtb0nQ7W6Yi1GoPKZLgA4LqkUbttByNxAHBrqPHM/h7w/wCF7qx8I6hbXi+Jr4X0ot5AxtbRQGit3APysHdiQcH5BmrSa7L4i8G6J/YV74Ygv9Jslsbqw1u0sQ7hWJSSKa5TDAhuV3jBBwOc0a2+7/g/5B1+/wD4H+ZwHiLwxeeG5bc3E1teWl2hktb6yl8yC4UHDbWwDkHgqQCD1HSsauw8a3mrtpumWWravoF0kTSSR2WixW6rbM23cXNugjJbA6M33e3fj6Q2bXh/wvdeII7u4W5tLCwslDXN9eyFIot2dq/KCzMcHCqCTg8V3Nv4Yjsfgv4pvDdaBrVtH9la0vrGFTNbyNOodWZ40mTK44PGM471heHprTXfh3e+E2vrTTr9NRXUraS9mEUVziIxtEZGwqMOCCxAPIyK3NM0K20P4T+MLO68RaHLquoQ2ssen2+pwyEJFNvb5w2xnxnCIWbjpkgFvt/T/r9CP6/r+vIwPCOjGfQ9bu7G88PXtx/ZNw81hf29w89tGpBMsTeX5YkGBtO88N9ccvpWmXetata6ZpsJmu7uVYYYwcbmY4HJ6fWu8+H+hzWdvrdzeajodvHqOg3NvbCXXLNHaSRV2KUMu5Cf9oDHfFYOh+HJbXxxp1hqPiDT9EZszLqlvfw3MdsVVmUl4pCAdygY3A8jjpl3XOubb/gseri7f1oi9a/DKTUNQuNP03xZ4bvL63jeSS3iuJ8kIpZgrGIK5AU/cJ/LmjwH5V34a8Y2d3Z2U8dvo0l3DJLZxNLFKJYlDLKV3rwTwDjnOM109oWudUum+Id/4Nv9EaGRp7yxax+1yNt+RovIAnL7sffGMZ3VjfCrVT4YtPE2vm40xHTS3htYby4iLSz+bE6gQlt7DCk5Axkdc1Ks1JS/lf32f/A0L0urd/8AI5zW/CDeH9KtbnUNZ04XlzCkw0tRP9pjVum8GIIpxzgtyORkEVztdr400m11KFfF+k39mYdSBnutPm1SGS7tZS+GXYX8x1JOQcbgPvAYyeKoV+otOhteFb7RdN1k3viKwk1GGCF3gtB9yWfH7sScg+Xnk4yTjGOTXW29zH4u8AeJL/XNE0nT49OVJLHUbCxjtD57OoFtiMASAoWPOWXbkmub8D+HdO8Sa+1vrWrW+mWUELTyNLcRQvPggeVG0rKm9s8biABk84xXReLrTU9YsX/0rw5o+haPAzWGl22u205IyAQFikZpJmzksQM4PI4FOekde39P17f5bkd/6+70POqdGoeRVZ1jDEAu2cL7nAJ/IU2nwxNPMkSFQ0jBQXcIoJOOWJAA9ycCgl7HrjaTpHhr4c6BNp2u+EVudSluZJ9R1HSZ7vzwjhFWMSWr7FA65VST0JFcPo+gy+OvGD2Eer6PaXtzMIrcm2kghuW+6PLSGHCA4B+ZU6885rqPEHh24vvh34Q0y11Pw+95povftUX/AAkFiPL8yYMnJmwcjngnHesr4V2Bt/iRpWo3d7plpaaZfo1zLdanbwgAE5K73BkHHVM042clzEzbjBuJQ1LwBc6RbXI1HXNDt9RtYmll0p7s/aAAQCudvl78HPl79/tnioYfBbJptpea3rml6GL1PNtob4ztLJGekm2KJyqnnBbGeoyOareIfD97pckl3dXOmTRzTsF+yarbXTknJyVikZgOOpGPzrpPEmnJ46u7TXtF1TSYjLaW9vc2V9qMNpJayRRLGcCVlDodoYFc9cEAiktY3/r+tjWaSm4rbX+vzZV0r4WavqviCXRxqOlW04tPttvLPO/k3cGCTJHIqFdoAOdxX06ggW4/hBfzR2U0Xifw21tqUvkWFwLuXbdTbipjQeXuyCBksAvzLzzXUeH7vTUMGg2+s6XKuj+HdQs5b6W+it4prm53sscRlZd6gkDcOMkngEE5FlcWkXh34aPJqOnj+ytamF6n26LfAGuY2DFd2dmEY78bcDryMis5Jf1u1+VvvI1s3/Xwp/nf7u55nd2s1jez2l2hjnt5GilQ/wALKcEfmKZBIsVxHJJCk6I4ZopCwVwD907SDg9OCD6EVreMpYp/HWvS28sc0UmpXDJJE4dHUysQVYcEEcgjg1kwQtcXEcMZQPI4RTJIqKCTjlmICj3JAHepg24pvcqVk2em+IB4f0rxBoljYfD3Tr8ahplpdyQRXN95zvLGGZYyJzjk8ZVvfNcj4/0Oy8NePdV0jS5WltbaUCPewZkyoYoSOpUkqfda9I8R6h4vju9Dj8J+NrK1s7TSLK3kig8WWsUSTJEFkBTzwDgjk4IPvXDfE+50268VQNps9nd3K2MK6ndWCgQXF5g+ZImAAQeMkAAnJ71U7c7t3l919Pl/wA6fJHHV3/h/wboWo/DTWtWvNe0uG9gntBHLKt5/oYcybkcJCQxbaMFQ4G3qO/AV3Pg77PqXw/8AFOgjULGz1C7ls7m2S+uUt0mERk3qJHIUN864BIzz6U/sStvp+av+FxLfU5i90hItTgsdK1G21uSfaqNYRzAFycBMSxoxbp0BHI5rcPw/ZdQ/stvE2gLq+dh08zzbxJ/zz83yvJ3Z4/1mM8ZqzpVvb/DrxhoGs32q6XqjQ3Syz2ul3IuGhQY5LqPL3cnADHkVG/gNH1ZpV8U6B/ZDMZRqB1KLf5ec5Nvu87fj+HZ14z3oVtA11+X6/wDA+85K7tLiwvZrS9iaG4gkaOWJxhkYHBB9wahrf8da7b+JvHWraxZIyW91cFog/wB4qAACfcgZP1rEt4HurqK3iMavK4RTLIsagk4GWYhVHqSQB3NRFtxTe45WTPUfB1p4Ok8H2moeMvDNikd5qUWl29zDc3UcjDbma4YGYqQu5PuqBknivOte0e48P+Ib/SLwfv7K4eBzjGdpxkex6/jXonjfU9O8P6b4f0KLSPDXiGw06zCR3S6o8zNO+HnO22uV2jecAuoJ28EjpmfFkwahe6Prwm0sahqFkI9QtNOv47oQzRYQMWSR8Bk2EZYng5JINVLe672/r7n9/oKO2va/9ff+B59RRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHe/E7/AJhf/bX/ANkrgq734nf8wv8A7a/+yVwVdmO/3iXy/I5MF/Aj8/zCiusn+Hmpx/DqPxpb3ljd6a0oiljhaTzoGzg7lZAMAkAlSRlhUafD/VW+G8vjaWe0h05ZxBHDI7+dMS23cqhSu3O7ksPuN7Z49r+R1rXVHL0UUUAFFFFABRRRQAUUV0emeGbTU/Aut66mozR3ejmEyWhtQY5FlkCAiXzM5GSSNnYc88AHOUVoaVZ6ddpfHVNV/s5oLR5bYfZ2l+0zAjbF8v3M5PzHgYrPoAKKK6Pw74Zs9f0LXLo6jPb3uk2bXot/sgeOWMMi48zzAVbL9NhGB17UdG+wbuxzlFFFABRVvTLS3vb5Yb3UYNNhIJa4nSR1HtiNWJPpxj1IrY8ZeF7fwvdaYtlqTalb6jp0d/HM1v5JCuzgDbub+5nOe/SjYOtjnKKKKACiustvCek2vhHT9e8S63c2SanLKlnbWOni5kdYztd2LSxqo3HGMkmubvks476RdMnnuLUEeXLcQCF247oHcDnP8RoAr0UUUAFFFFABRRRQAUV1zfD2WzW3TXPEWh6Ne3ABWwvZ5TNGCAV8zy43WLIIPzsuAecVmeK/CmoeDdYTTNXa3a4e3juP9Hk3qFcZA3YwT9Mj0Joem4LUxKKK1LTw/dXvhnUtcieEWumywxTIzHeTLu2lRjBHyHOSO3Wjpf8ArsBl0UUUAFFFFABRXceGfhddeLdNa60bxFojMhije3lNzG6yyA7IsmHYXO0j5WI4645riHRopGSRSrqSGVhggjtRs7AJRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHe/E7/mF/9tf/AGSuCrvfid/zC/8Atr/7JXBV2Y7/AHiXy/I5MF/Aj8/zPX/D2vWeifDnwnFrTMNG1O41Ox1HapYrE/k/OAOcowVhjn5eKsalq+l674H8Q6dpdy3/AAj+mXGj6faTFSm6JTNvlIPQsxduQDyM15tP411q48NQ6BM1g+nW6lYYzpltvjzjLLJ5e8Mdoy27JxyTTB4v1dfC83h5Hs49Nn2GVI9Pt0kkKNuUtKE8wkEnktnBI6EiuVtO9/L8Gn/n951RTUeVef4tv/L7jqvGfjHxL4Y8b6houg6leaLpml3DW9pYWkrRw+Up+VmQcSFhhizZ3bvStxPE58C/FLTnhuJ9KsNZ0+2k1uxs5WgSCSeHDsFX7jLu8wd1JwMVwlp8Q/Elnb2kSXVrMbJAltNdadb3E0CjoElkjZ1A7YPHaufvLy51C9lu7+eS5uZmLySysWZ2PUknrUrS19f1/wCH3+Q3re2n6f8ADdD1dLrxN8NPDfi65vNb1GO+vtQbTbQi6kHmyZDzXeM8tsCAP1/eVJPq1p4T8BeF4tJ8S65oEGo2P2qebRtOjk+03AkYOHm+0RsSmAuzGAMdc15tr3izXPE0FhDrt+12mnQ+RbBkVdiYA5KgbjhR8xyTgc1Lo/jPXdD0ubTbG7iewmcO9nd2sV1CW/vBJVZQfcDNHS3p+H9X/DzB6u/r+P8AVjV8eeItG8R22mz2k99faxEJEvtQu7CK1N0nHl7lSVwzr8wLHBI25ziuNrW1vxRq/iKO2i1W6V4LUEQW8MEcEUWcZ2xxqqjOB0HYVk0htne+G7mbw98LtS8R6C/l60dTjsXukUGSzt2jZtyH+Auw27uvy4BGa2tO8Ua94l+CXjb+355NQFoLFIr64G6Xm4BMZkPLDjIBJx+Ned6H4i1Xw5cyzaPdeSZozFMjxrLHKh/heNwVYexBrYufiZ4qutHuNJlvrYabcQ+S9lHp1skIXOcrGsYVGzzuUBsgHPApvV/1/X/D/fNv6/r+tPu2PhzruryaR4m0eTVL19Mj8OXrpZNcOYUbAORHnaDknnHc1yfhR2i8W6ZMmkNrXk3CzNp6IXNwqfMy4AORgHsR68Vf0z4g69o9r9n04aTChgNs7f2JZs8kZGCruYizg453E575qrD4y1m18RW2uWMtpY6jaqVhlsrCC3VcggnZGgUnDHkjPvwKd2p8y/rVha8Wn/Wx6jonjTUPEfiW5tvDXjLVr28voZlt9J1/T0Fq2Y2yiNHMRGwXJUhEGQAeK5b4RQ6dcSeLItbuZbTT20FxcTQpvdE8+HO0etYk3xI8TzfaGF5awz3UZjmurfTraG4kU/eBmSMSHPf5uay9O8TappOj32mafJBFb6ghjuT9kiaR0OCU80qXC5UHAYDIzSWl9OjX4P8AzKvdq/e/5HT/ABI1vxDDeR+H5bueHw7BCg0y0gu3ktprYcxyAk/vM9ckcHgBcYHB1tXHi3V7rwxb+H7h7R9Ots+QhsIPMiy247ZdnmDJHOG56HisWklYL3Cu4+Jf+p8Gf9ixaf8AoctclpmpT6Tei6tY7WSQAqFurSK5Tn/YkVl/HFbWqfEDXtasxa6l/ZU0awC2jI0WzR4ohnCo6xBkAycbSMZ4pvb+vMlfFf8Aroc1To5HikWSJ2R0IZWU4KkdCDTafDK0EySoFLRsGAdA6kg55Uggj2IwaAex6n4l8c+LYPhf4JuoPFGtR3N0L/z5k1CUPLtnAXcwbLYHAz0rG+EOt6rB8XNG8nU7yP8AtC/QXmy4cfaQWJIkwfn5JPOetZsvxK8RT2dvazroslta7vIhfw/YFItxy21TDhcnk461Q0Pxhq/hu/e90c2MN00olEz6bbStE3PMZeM+X16LgU4e7JNkzi5QcV1NTW/iN4ju2v8ATYbpLPSJVe3Gl28Ki2RC2eEII35GfM+/nndWt4v8Q6z4NvLDRfCOp3ekaWmn21xG9hM0JvGkiV3mdlILksWHJIAXAAxXIav4mv8AXIFivodMQK/mbrTSra1cnBHLRRqxHPQnH5VZ07xvrumaZDp0M9rcWlu5eCK+sLe7EJPJ2ecjbOeflxzSWkbGs2pTcltr/X6HrXhXV72yurDxBFILXVdT8L6hd38CKFjuXh8wQzyRfdYsBnJGCQT3NMsPHniR9J+G9xJqcj3Os6pLBqNwyqZLqJLlFWJ2xlkAlf5Txz7CvLbX4heJrTWL3VFv4p72+i8mea7s4bgmPBBQCRGCqQcbVwCOMYFV08Z65HYadZpcwLDpd2byyxZw7oJS28lW2bgpbB2528DjgUL4k3t/wW/yZFtGv6+FL80Q+LbaGz8a63bWsaxQw6hcRxoowFUSMAB+AqHw7NbW/ijS5r4qLWO8heYsMgIHBbI9MZqtqF9capqVzf3ziS5upnmmcIFDOxLMcAADkngACq9FP3LeQ5+/fzOl+ItvdW/xJ8QC9Db5L+aZWJyHR3LIwPcFSCPasjVU1ZJLY62t4rtbRm2N2HyYMYTZu/gwMDHHHFa1h4/8Rada2tvFdW06WQxate6fb3T249EeVGZB7AgCsfVtX1DXdTm1HWLya8vJjmSaZtzH/ADoAOBSS5Uoobd22ynXpvhfxb4js/g14m+ya/qkH2K6sI7Xyr2Rfs6MZtypg/KDgZAxnArzKtfQvFOr+G/tS6RcpHFeRiO5gmgjnimUHIDRyKynHbI4yar7Lj3t+DT/AEsLrc1/DM1x42+I2gw+M9SvdRt7i6jt3lvbl5Cybs+WHYkgEnHB43VZl+InjRPEjW631ykazmAaGAfsgG7b5H2b7uP4cYz+PNYWveMNd8SrbJrF95kNrn7PBFEkEUWepWONVUHgcgZrRHxM8VBxL9vt/tYj8oXx0+3N3txj/j48vzc477s+9CdreQd/l+v9fIg+IemWGjfETW9P0gBbOC6ZY0VsiPuUz/sklfwrm6V3aSRnkYs7HLMxySfWn2872t1FcRCNnicOoljWRSQcjKsCrD1BBB7ioirRSY5O7uep3FpB4e8C+FtLTxRpmian5w166juo7pn8xsfZ/wDVQuBiMZwSDl+lc98V9Ntbbxq2raVJFNpuuwrqNtLCG2EvnzANwB4kD8EAgYyBWH4j8Xav4suY7nXJLWadOPNisIIHYYAAZo0UsAFAAbOO1JrvizVvEdpY2uqSW32fTw4tYbaxgtkiDkFsCJFByRnnvn1NVLV38/w/pL7hLRW8vx/q/wB5jUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB6B8UhGP7L8tmb/XZ3Lj+57mvP6734nf8wv/ALa/+yVwVdmN/wB4l8vyOTB/wI/P8wor3LQZT4o+CWi+EdRleafUpL4aZJK5YxXEHltFGCeispkTA/vCjWLZ9G+COp+CbJSLy1m02TUApwZLq4Z3Mbeu0LCv1X2rkaav8vxdv6+Xc6ovmjf1/DT+vn2PDaK7XVdM8G+GdXk0TVoda1K9tH8m9u7K8igiSUcOscbRMXCnIyWXcR0ArT0bwN4eXx1H4c1uW/uk1O2S50e9tbqO2WYSRl4ldXicqWOEyDwexpf1/X9a9B7bnm9Fdx4c0Lw1d6X4l1HXtP1iG30gBoxFqMSNvd9sduwaA5YkMS4wAFPy1p6d8NVh8J6bqV94b8Va/danELmNNEj2QQQkkDfKYpMucZ2gDAIz1oB6Ox5pRXZePPA58L22m6lbWuqWdjqQkVLXWLcxXMEkeNytwAyncCGAGc4wCK42gdgq8mh6tJosmsR6XeNpkb7HvVt3MKNkDBfG0HJAxnuK2vD/AIe01vDd54l8TPdjS4LhbOGCyKrLc3BUttDMCEUKMliD1AANdXpy+GZPg141ufDZ1K3mKWKXNnfyJMU/0kEMkiKgYHkEFQRjvmgm55paafeX6zmxtJ7kW0RnnMMRfyox1dsDhRkcnjmq9dx4KttL1HRdeghOtWOpW+i3VxLcW2pKkFwqYPlPF5W4oeMgyHOPy5fQNOg1fxFp+nXl7HYW91cJFJdS/dhUkAsfp+H1FO15KK/rVhdJNmfV+20LV7zSrjU7PS7240+2OJ7uK3doojwcM4GF6jqe9ejXngLw3pl1errvh7xto2n26P5WrXASWCRwp2btkGFDNgZDv1/EZXwq0681dPF2n6Zbvc3dzoEkcMSDl2M8OBSWqduib/Bv9CrWav3sef0V1Hiix8L6NAmmacL671mFFW9u1vka0WYH51jTyQzDtnd16bhyeXpJ32FsS21rcXtzHbWcElxPIdqRRIWZz6ADk1f1Xwvr+hQRza5oepabFI21JLy0khVj1wCwGTVTT9Tv9JuvtOlXtxZT7Snm20rRttIwRlSDgjtXpOh2RsfhP4pubfXYfEAvLdBNpdozn7GfMRjcyrKFbKkbdyKwy3LYpv4W+wL4kv6/ryPLaKKdGUEimVWZARuVW2kjuAcHH1waANLSfDGv69FJJoWh6lqSRna7WdpJMFPoSoOKp31heaXfSWep2k9ndRECSC4jMbpkZ5U4I4INeo+I9Q8LRfCfwZE+ka4bGd76WO3j1qJCHEoUs5+zEOfQ7VwOOa5HwHpGg+JPHVpouq22oi31G5WG3e2vY0e3BJ5YtEwk4x0Cf0oXvOyE2oxuzlKK63VrXwJp8d5YWlxrd7fQxsI9QTylt5JgeF8llD7MZG/eD32dql1DRfDXhRLWz8SQarqOrTQR3NxHY3kdtHarIodEy0UhdtpBP3QM4560LVXKaafK9zjaK9R0H4feF77VEmv7rVX0K/0qfU7O6geNJoBBu82KRCjB2BXGQVB698Ca28HfD24tPDV5v8SJH4lu2s7a3M8Ba3ZJfLaR38vDD50woUH73zdKOtv63t+aJurN/wBbJ/k/6Z5RRV3WdObR9dv9NeQSNZ3MluXAxuKMVzj8KqwTy2txHPbSvDNE4eOSNirIwOQQRyCD3pJpq6KaadhlFe867eeP73XvDMmieNEsI7jRrB2F74hiiDSNCCzPA8m58nrlDu9681+KtvBbfFDW47Oxexh88FYmhMQJ2jc6qeis25h7EU5XjJxfdr7nYOl/T8TkKKK9N8L3Hhxfg34m+16Vqkuy7sBdeVqcaea+ZtpTMDbAOcg7856ju/suXa34tL9RLV2PMqK3rbS7TxR4q0/SfClldWZvJFhVb+8WchyeWLLFHhQO20ng89q0mi+HyaodOP8AbzRBvKOri4i2Z6eYLbyt2zPO3zc49+KLNgcfRWn4j0K58M+JL7Rr4q01nMY2dPuuOzD2IwfxrMqU01dDas7BRXpPjLxLrHgbWP8AhFPCOoT6PY2EMBkeybyZbqZold5JHXDMSWwATgAAAVz3jDxxc+MrLR0v7SGK506CSKWeIBRcszly5QABW55xnJye9N+X9eYlqjl6KKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA6DxTcTT/ZfPmkk278b2Jx931rn63PEn/Lt/wL+lYddGJ/isww/wDCR2914i0Fvh5oemWOo6xb6zo8013HIllGsZlkKHAkE+5dpThtuT6CltvFmm2vgXWbd9S1e88Q6rdWt40s9qnlpJC7k5lMxZ8h85KDkYxzkcPRWF3+X4Wf6L7ja2nL01/H/h39522qap4K8Tau2t6xJrenXt25mv7SytIpopJTyzRyPKpQMcnBVsZ7isjxF4qm1jxJBqdjEbCOxjhgsIlfcbeOEARjd3bjJPqTWBRS2tboPe9+p33j7x1pPiLSYbPw9Y3FiLu7bU9WWUKFku2RVOzBPyD5yM4++eKqf8JB4d8Q+GtPsPFo1K01DSofs1pqGnwxziWDcWCSRu6crkgMG6HkcVxlFGm39f1bQDW1seHEjto/DjapM6g/aLi/WOMOeMbY0Lbe/V2z7Y5yaKKAOo8P+IdNXw3eeGvE0d2dLnuFvIJ7IK0trcKpXcFYgOpU4KkjoCCK3LfxF4H0v4feIPDumf25LeatFC39oXFrEoMkUm9Y/KEp2IcYLbnPOccYPndFArHdeEdV8H6Bbag95qeuPcalpU1hLHFpMJSFpAMsGNyC4GO4XPtWRpr+D9P8W2ct9/aetaEqMbmKS2W0mZirBQoWZuA2053Dvx685RQ9Xca0Vj0Dw74h8HeD9bl1nR7nX71/Ikjj026tYY4pN6ldssqytvQZyR5YzgdKzfBviXT/AAzo/iCRrnUE1XULB7K2W2hUImXjfeZfMDA/IRgKeuc9q5GijXX0t8v6YHXeKNa8P+JdLt9TkkvovFBjxfbLKNba7fd/rCwl3K+3liEwx7Lya5GiigDR0RtFGo48SR372TIRusJEWRG7NhwQw/2cr9R36e11/wAK+F9F1ZPDJ1bUdT1S1ksTNqFvHbxW8EmN+ESRy7kDGSQBnOK4eih6qwdQp8IjaZBO7JEWAdkUMyrnkgEjJ9sj6imUUAeg6rq3gjU/COgaH/a/iCL+xvtH77+xYG87zZA/3ftfy4xjqc+1ZngHVvDvhzxlaa3q93qezTrtZoIrWwjkM6gn7xaZdh6dN31rkaKFo7oUkpKzNnXI/DuDNoep6ndzySkvHeabHbqqnJyGWeQk5xxgfXtW1f634Y8VR2l54lm1bT9Whhjt7iWxtI7mK6SNQivh5Yyj7QAfvA4zgZxXGUULRWKk3KTk92enaT4+8N2161tKmqWWjWekXOl6fFFbx3ErmcN5k0pMiANlgdoyO2RjJzrbxfocGi+C4CdQa58N6pJcSr9mTZNC8yyZDeZkOAgG0jGT97jngqKFo7/1vf8AVk2Vrf1tb8kjS8SahBq3irVtRsxILe8vZp4hKoVwruWG4AkA4PIBP1qhAImuIxcu8cJcCR40Dsq55IUkAnHbIz6imUUkklZDep6B4m1XwN4kvtOnfVvENutnp1tYlRo0DFxEgTdn7UMZxnGDj3rE8c+JLPxHq1l/ZNvPDp+m2EOn2v2pgZpI484Z8cbiSeBwBgVzVFN6u773+b/4dh/wwV1fhbX9HtfDOu+H/EIvorbVGgljurGNJXikiLYBjdlDKd5z8wPArlKKOjXf/hwOrOt6D4b1rSNS8Drqct5YTCeS41QRospGMIIkJ2r1yd5Jz2qdpvh2+pnUSPECxE+adIEEOzd18v7T5m7ZnjPlbse/NcbRTu0Bp+I9dufE3iS+1m+CrNeTGRkQfKg7KPYDA/CsyiipSSVkNu7udle694d8WQ21z4rk1Ox1eCJIJrywto7lbxEXarMjyRlXCgAsCQcZwKyPEt34dmltoPCmnXVtbW6ESXV7LunumJzuZQdiAdAF/EntiUU3qJaBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAG54k/5dv+Bf0rDrc8Sf8ALt/wL+lYdb4j+KzDD/wkekR/D3SNT+DaeJtJkvk1uLzZJrSWZJIpYYWRZXQBFKkeYrYJPAPWmTfD3TLD4LP4ov57v+3JHhkhtUkVYo7eV2VHdSpZi3lSEYYcFT253LBtf8LfDLwX4hg0K9urSzu7+W7RrdxFJbSiNTvbBCo6lgGPHcZxUtxf6jrXgDxR4m1Pw/dWWjy3uk/ZIGiYwi1iMiBEcgBgBtBI4y3bNZNb2f8AL8rtJ/n+D8jWL9xN/wB752bt/Xmjz22+H/iG6t7aUW9nAbtA9vDd6lbW80ynoVikkVyD2IHPbNM0rwL4g1m9vbOztIY7qwDG5t7u9gtZIgoJYlZXU4AGSRwO9dN4y8F+JvFPje/1rQNNuta0zVbhri0v7VDJCImPyq79IyowpV8bdvpSa/4tj0L4paNqljPHfz6TZ2trfzxMHS7dIgkw3fxAqSm7vjNStbdL/h6+mz9Snpe2tvx/4foc1pPgbW9ca+GmDTpfsG83G7VrVAirjc43SDcgyPnGV96bpXgjXtZ0j+1bS2t4rDzPKW5vb2C1jd/RWldQx+ma7bxjo1r8O/Ceo22nTrK3iq6DWUiNnGmJiRDnsXZlB9fLNWpI28T+BvDlx4b8H6f4nOl2As7y233TXNrKJGJbyoZ0yj7twYKecgnijpf0/r5bev4j3+/+v19Pw8x1zw/qnhvUBZa3aNazlBImWDLIh6MrKSrL7gkcGs6u38dSa5BoWi6ZrXhqx8PW1s00lpbRSSmYbypfekszyIMgEAhRktjPNcRSG0aeh+HdV8R3MsOj2vnGGMyzO8ixRxIP4nkchVHuSK7ZPAkmm/CnxDqOu6JafaLZbaXT9Wtb3z1k3zqjruilaJsDtjI3fSqfhu1l8RfC7UvDmgp5mtjU4757VCBJeW6xsu1B/GUY7tvX5sgGt/RvBvibRfgr44TVbG5tmuI7O4j0+RT5wSOfLzNF95FCg/MwHAJ6Cm9NPT+v67P5R/X9f11OP8N+G5rjQ9V1KXSLLV4Rp0zxqutQwzWRUj/SDAH8xguD8pXB3A+lcxa2097dw2tpC89xM4jiijXczsTgADuSa9I+Gvg/xN9n8QXX/CO6t9nvPDl0ltN9hk2Ts6rtCHbhie2OtYPhjwt4ss/iJpOn29tLoWtljc2japC8AXYGfcQy9PkI6Y9e9PTnSe3/AAWPXlbX9aIUfCrxc01xDHYWss9qjPcQQ6naySwqASS6LIWXgdwPSl8Gafpeq+HfFUWoaXBNcWGlPfW155kqyRuJI0C4DhCuGJ5UnPftXa6V4e1K5166fxJ4JXwjb+RM83iPTpbm2S3+Q/MHaVoXVs7SiD5gxxWL8Ilk02x8W69eaK2o6Xb6M6SCVGEEriaFvKZwMZI529x7UlqpJv7L++zL0urd/wDI46/8I6vpmhWusXyWcVpeRrLAP7Qt2lkQnAYRBzJjOQfl4wc4rErvPiB4d1e8YeNbaLUL/QNUUTRXk1uR9lG7Z5MmBtXacKpGFYY2+g4Okr9RadC3pkunwXyyavaT3lsAcwwXAgZj2+Yo3HqMZ9xXTfEXTNK0+58Pz6Hpy6dDqOiQXssCzPIBI7yAnLknoo9uOlczpmk6jrV6LPR7C61C6YFhBawtK5A6naoJr0X4m+EfEi6d4Yum8P6qLex8NW6XUxspNluytIWDnbhSARkHGM05bJ+f6MlfFby/yPL6dGhlkWNSoLEKCzBRz6k8D6mm0+GGS4mSGCNpZZGCIiKSzMTgAAdSaAex6ongW20HwLo97NpPhzWNT1KWdpZNR8SQxQwpGwVVj8u5jEhPUkM2OhAribLw5qPi3xJcWegWOmW9x5gRLKLUo0Qt0xE00pMmSCflZuvpiu18S+BvFs/wv8E2sHhfWpLm1F/58KafKXi3TgruULlcjkZ61k/CDQNZuPipo1xb6TfSw2Goxi7kS2dltiCf9YQMJ0PXHSnD3pJMmbcYNoxJvh54ottKk1C40wRRRQG5eJ7mITrCDgyeTu8zZyPm24wc5xUGn+Ctb1HTItQjitLa0nYrDLf6hb2gmwcHZ5zrvAPGRkZ4qTxZoPimy1C51TxPpOr2wurlh9r1C2lQSucnG9xycAn6Cul8XeHNa8Z3mn614Q0261fSnsLa2jSxjMps2jiVHidFyY8MGOSACGyCc0lrG5rNcs3Htc5/Tvhz4q1TXLnRrPS86lbIJHtZLmKORkIzuQMw8wYGcpkYwe4q8vwg8buyhNIiZXIEcg1C2McrEsNiP5m13yrfKpLDHSvQ/Den6gZbTRraCa/1jRvCuo299JZgzeQ8vmGG3Lrn5wOAuepwOlYun2d8nhf4Uym2uFt4Nenjmfy2CRyG6iwrHoGO1sA88H0NC1ko/wBbtfp+JF3Zv+vhT/U8jlikgmeKZGjkjYq6MMFSOCCPWm1u+OFKfELxEjKVK6pcggjGP3rVn6PpF9r2sW2l6RbSXV5cvsiiiUsT3JwOwAJJ7AE0oXkl3KlZNnpOj+AvCXiDwzYy20Ws6drGsi6XSYJ9QinjmaBM5bECEBnygGeo615SQQSCMEdQa9m8Y3dh4d+IWi6Z/wAIt4le90FILTSjFerbi7Mb8SRxNbOW3ybuQx3fpXHfFrQDoXxAuXS0msrbVI01K3t50KPEs3zFCpAIKvvXH+zQ9XzLZ/0vv/AEmlZ7/wBfl+JxVdxofw5k1fwBqeufa7FLmCW2W2R9YtI02Sb9/mhnBjb5VwGKk88Htw9d/wCCtOu9d+Gfi/SdHt3vdSeexuI7OEbpZI0Mu9lTq2Ny5xk8iq+xLvp+av8Ah+Alucdqmj3OkXaW11JZyyOoYGzvYbpcZxjdEzAHjpnNbf8AwrfxPvEX2O2F2U3iwOo24u8Yzj7P5nm5x/Dtz7VpaHoV58PvHPhvVPG9ktjbfbEkaCV0aaNVI+dogSygE55AztNNl+HHjV/EjXC2Vw0bTG4GuB/9EK7t3n/afu4/iznPtnihJOwd/l89/wCvmcO6NHIySKVdThlYYIPpSV0nxD1Ow1n4ia3qGkENZz3TNG6rgSdi+P8AaILfjXN1EW3FNjas7Hf6ppXgfSfDnh7VJdK8QS/2zBLKyLrEC+TslaMgf6Kd2due3XHvWL458N2fhzVrL+yZ55tP1Kwh1C1+1KBMkcmcK+ONwKnkcHg16Lf6b470v4f+CYdF8J3FzNBa3DTCfw4l3JCxuXZeZImZMgggDGc5rm/i4k8z6HqXiCBLPxVfWztq1ojn92FYLCxQk+WzJ1QYAwOBmrnZSaX8zXyuwX6f5HnNFFFIQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAG54k/5dv+Bf0rDrc8Sf8ALt/wL+lYdb4j+KzDD/wkFFb1z4K1+08Iw+KJbJG0aaQRrdRXMUgDn+FlViynj+IDHTvTIvB+uzeDpvFSWONFhmEDXTzIuXJAwqFtzfeH3Qe/ocYd/I3MSiiigAooooAKKKKACiityx8LTah4Q1PxBbahZlNLMf2m0bzBMBI4RWHybCCT/ezweOmQDDorQ0rSf7VS+b+0LGy+x2j3OLybyzPtIHlx8HdIc8L3waz6ACiitzRfC0uu6LquoWmo2SSaXbm5ms5fNErxAqpdSEKHlwMFgfbFHS4dbGHRRRQAUVp+H/D9/wCJ9Zi0zSo1aaQM7NI4RIkUZZ3Y8KoAJJrV1LwLcWmh3WraZrOk65a2MgjvP7OlkLW+TgMyyIhKE8BlyM96HoC1OXooooAKK6Wx8FyT+HLfXNU1rS9FsruV4rU3zTM9wU4YqkUbttB4JIAzWFfW0VpfSQW97BfRoRtuLdXCPx2Dqre3KjpQBXooooAKKKKACiiigAorqz8M/FB0O51iC1srqwtVZpprTVbWfYFXc3EchOQpyQBnFcpR1sHS4UUVdt9HvrrRrzVYIN9lZSRx3Eu9RsaTdsGM5OdrdB25o6XApUUUUAFFFFABRXQ6d4F8Qapo6arDawW9hI5SK5vr2C0SVh1CGZ13+ny55qt4g8J654Va2XxBp72TXSs0Id1JdVbaTwTxnoe4wRkHND03Dcx6KKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA3PEn/AC7f8C/pWHW54k/5dv8AgX9Kw63xH8VmGH/hI9m8MapZQfC/w3omtzpBpGvT6lZXMshwsDEwmKY9hskCnJ6AtVjWLizvvh5rXhbRryGbS9HuNJsIrqJg0csrNM00wI4ILsefRRXm114z+2eDbPw5LoGleTZb2gula585Hfbvf/XbCTtHBXaOwFMi8Xtb+C7zw3baNpsMV60L3F4pnM8jRMSrcylAfmYcIBg9M4IybTun/d/Bp/5/ejWN4wst/e/Fv/gfczc8T+IF8K+LL3QdE0TRlsNKuGtcXulwXMtyUbazySyIXyxBOFKgA4GMZrcsX0Xwz8SNMtZNO09dE8R2NvcNFeWMN02nvcRfKUeVWOEchueCvUGuSbx+908FxrXhzQ9Yv4FC/br2KbzZcDCmQJKqSEAAZdSTjnNc/rGsX2vatPqWqTma6nILvgKBgYAAHAAAAAHAAqVdWv8APzX9a+VtCnZ3tp28v6/HqehaaLvwl4e8X6j4g0zSJZ4r06daRXGkWsg+2k5dk3RnaqIpIUYTLrxVs2WmeGPB3h9LTUPCtnd6pZLf3FzrWly3szlnYbE/0eWNUULjA+bOc44ri/Fvj7V/Gdrp8GrJbItihG63QqZ3KqrSyZJy5CLkjA46U3TfGtxaaAuianpWm63p8TF7eLUY5N1sWOW8uSN0dQTyRuwfSjW33fh/m9fwDd39f6+7/MvePH8O3Vtpt7o93pcmqyCRNRi0e2nhtuMeXIqyxptYgkFVGPlzxmuNrW1vX11iO2ih0fS9KgtgQkdhAVLZxy0jszv0/iY47dTWTSGztvD0Nrofw7vPFjWNpqN++orpttHewiWK2zEZGlMbfK7HgDcCBgnBre07xENe+DPjYXGkadaXluliGurC1S2Eym5GA8cYCZBBwQBwTnOK4Xw/4ou/D8d3brbWl/YXqhbmxvYy8Mu37rfKQysMnDKQRk81tv8AEy5/4RXUPDlr4d0Kz0q/jVZILeKYMHVtyy+YZS7ODjG9mXjGMEgt7k9P6/r+rljwFeDUNI8Q6XfWOmT29roN5cQu+m25mSQAEN52zzMjJx83H4CuS0CbTLfxFp82vwSXGmR3CNdQxHDPGCNwHTt7j6itzQ/HUOgWs0Vn4T0OSS4s2srmeV7wvPGwAfIFwFBOOqgY7YqtpvjF9E8W2ev6FoumadNaIypbx+fJC5ZWUswllZicN03Y4HHXLu1Pm/rdha8Wv66Hodje2up6zex6NB4H8SxSQSm20i30c2VyBsODG7wDc6/e2l3J2nHPI534RWEOqSeLLK6vodPgn0F0ku5/uQr58OWP09Ky4PiHJYXct/o3hrQdL1N0ZFvrWKbfFuGCURpWjQ4JGQgxnjFZei+KZdC0XVbC006yeTVLc2015L5plSIlWKKA4QcoDkqT74pKyvfX3Wvwa/Xfcu+q9b/kbPj7UJNPvX8LW+jWen2mnKsIlfT7cXdzt5EzzKm47hgja2CMZLck8VXQ6j4vfVfC9jo19pGnySWEXk2+pEzm5RN27bnzdhHUAFcAE4APNc9SSsK9y3psuordG20eS6E94htjFaswadWIBjIXlgePl712l8tt8O/Dmo6J5qXfibV4Bb6gI3DRadDuDGHI4aUlRu7LjHXNc/4P8XXXgzVZ9QsLKzupprZ7f/ShIDGGxlkaN0ZGwMbgQcE1LN4tsnsryC28HaBayXcRia5X7VLJHkg7k82dwrcfexnr6mnLbT+vL/P+rkd7/wBepzdOjcxSLIoUlSGAZQw49QeD9DTafC6xzI7xLMqsCY3JCuM9Dgg4PsQfegl7HqfiXxhqUPwv8E3KW2imS4F/vD6FZMg2zgDahiKp77QM9Tk1gfCy7+1fFDS7S8stOurbU71EuYbnToJUKliSFDIRH1/g2+lRXfxCt77SNP0y68GeH3s9N8z7LF5l8PL8xtz8i5ycnnknHas/wx4uHhXWl1W00HTLq7inE9s901wRbEZwFCSqCP8Af3dKcLKSbJmm4NIu6z4xtW+3aXpnhfRLXTSjwxK9oslxGd3Ev2jiQuMdMhOcbcVoeJr9fA15a6Domm6VIYrOCe5vb7ToLuS6kliWQkGVGCINwUBcdMkkmuY1nW7PVYgLbw5pmly+ZvaazkuWZ+vykSzOuOc8AHjrV6Dxs7abaWetaFpOuCyUR201+swljjHSPfFIhZRzgNnHQccUl8Nnv/X9f0jWbTm2ttf6+78z0fwvbaQLm08QHw/pUkGreH7+9uNMntElhWe2LgNGWBaNWIB2qR3HQDCWWtWE2m+A7uXwp4ba58RajJa37jSogphSdUComNqEiU5ZQGO1eeK4az+JupW2s3Wo3Gl6XeNNZNp8UEscscNrbMCGiiSKRMAhj1ye+c5JqReO7uHTNDs49M08f2DfNe2U2Jd6lpA5jP7zBQlV7bsD73XIviV9v+C3+TX3EWdnrr/9ql+af5mX4msYdM8W6vYWilYLW+mhiBOcKshA5+grLq3q2pS6xrV9qdxHHHLe3Elw6RAhFZ2LELkk4yeMkn3p2j6iuk6xbX72Ntfi3feLa7DmJyOm4IykgHBxnnHORkFQVklIqTu20ez29nfeENR8GaXFPo6WWmwmbWILnWbSB3kuh+/R43lVvliKqMjtXkni/QH8LeMNT0WQ7xaXDLG+c+ZGeUbj1UqfxqTxB4o/4SPxQ2uXmj2EU0svm3MELz+XcsW3EtulLDPTCMuB0x1pPFfimfxbqNteXNjZ2RtrSO0SO08zBSPIUsZHdi2MDJPRRTd373XX+vl0EtNPT+vn18zDr03wv4mvrX4NeJvKg0tvsl1YRx+bpNrJuDGbO/dGfMPHBfJHOCMmvMq3fD3iy48P2WoWBsLHU9P1JUFzZ3yOUYoSUcFGVlYbmwQw6mnvFx72/Bp/pYNncuaBE/j74gaNp2qm1tY7qdIHaxsobUbM5OFiRVLHkAkE9PSrEnjpU1dol8M6CNIV/LGnnTYS/l9MG42+dvx/Hvznn2rN1fxdcalJYmx0zTNEWwfzIBpduYmD8fOZGLSMeB1Y4xWj/wALEc6h/abeGPD7av8AeOoG2l3mT/nr5XmeTvzzny+vOKE1pdBrr8vlv/wPuM7x1oVv4Z8dato9k7Pb2twViL/eCkAgH3AOD9KwKmu7u4v72a7vZWnuJ5Gkllc5Z2JySfcmoaiKaikxvV6HbfFvcnj57ePiyt7G0jsVX7iweQhXZ7Elj9Sa5i/1zUtUsNPstQumnt9NiMNojAfukLbiucZIye+cdBxWraeNJV0q20/WdH0zXYLMbbVtQWUSQLnOwPFIjFck/KxIHYCq3iXxXf8AieW2F3Fa2lpZoUtLGyhEUFupOW2r6k8kkknuap/r/XzEtvl/XyMSiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKANzxJ/y7f8C/pWHW54k/5dv+Bf0rDrfEfxWYYf+Egor2XStG0zxP8ABPT9ObTbFNfuPtktheQ2kcc0zW2xvJZlUF90bP8AeJ5UUzUvD+maJ8DbvTV0y1k8Qo1hc3d01urTxG5Z2WFWI3JhI48gEfePqaxatf5fi7I2j7yuvP8ADc8dorsLzwp4e0O+GmeI/Et1baqmFuorPTBcQ2rnqjyGZSWX+LarAHIGcVc0b4cW134vl8N6xrM1revB9osHsrJbmK+j8oyKUZpY8FlHygjBPBINIZwdFdf4c8KaFrdlrt5d65qNja6RGZjL/ZaSCSMsFQH9+NsjMcBORwfmpsPhLSLDw/Zal4r16fTptRTzrSys9P8AtUrQ7iolbdJGqgkHAySQM0B5HJUVv+JfDMeiwWWoaZqKarpGoB/s14sRibchAeN4ySUYZHGSCCCCawKACiuh8P8AhiHU9LvNY1nUf7K0ezdYnuRAZnllYErFGmV3NgEnLAAdT0rurG3sYPgX4xGh+JrjVdP/ANCJsbmBrd7WU3C5by97p8wA+ZWP3cHoKNhXPJKK7nwRaxS6J4gOj+IL2y1L+xLlr22bS4nhmt1ILRiUy7gWwvIQEc8+vK6Do8/iDxBYaRaPHHPfXCQRvK21VLHGSfTmna8lFbhdJXZQorvF8F+EX1m70pvGtxaXdqkhLX+j/Z4pWRS21WM2QSRgb1XOfXAMvw1utRuPD3jTSLae6lt5dEkkSyjdijzGaFQwQcF8HGcZ7VO6bXRN/cm/0K5XdJ97Hn1FdP4k8N6R4ctIraTWbqbXvLVrrTxYp5Vq5+9G0wmOWHoF9jtOQOYpiCiremancaRfLd2QgMyA7TPbRzqPfbIrDPocZHau0+LF5cajP4TvL2Vprifw3bSSSN1Zi8uTQ9r/ANdRX1t/XQ4CiinRhDIolZlQkbmVdxA7kDIz9MigY2ivZp9S0jwv8LvDC6D4s8R6NDqEl3NLcadpkcc12ySBB5m26UgKOAAzA9eDXDeHNLtvH3jtbDWfEGpi61G4WG3vZbQXLzEnAMu6YFeAOhf07ULV2Qm0ldnJUV1uqeF/Dmkw3dtN4vSbV7aNmMNvYtJbO4P+qE4bJcjvs2ZGN3emy+FdH0aztD4s1u7sb67hW4WysdPW5aGJxlDIWljClgchRuOCM4zijdXKaadmcpRXomjfC+z1LWzaXXiQW9hcadJqVjqcdkZIZoY93mbwXVo2XaQVAY5z7E2IPhj4buLfRrqPxvILbXbg2unbtIYSPKH2NvXzcIgJT5gxJ3fd4NHl/W9vzJv/AF8r/keZ0VZ1Kwm0rVbvT7rb51pO8Em05G5WKnH4ioYJmt7iOaMIXjcOokjV1JBzyrAhh7EEHvSTTV0NprQZRXuGsab4um1bQH8KeBNN1CyvNKsp5SPC9qbeWV4g0m6XyhtyTzhlx2xXmvxFstK034iazZ6AqpYwz7URCSqNtG9VJ6gPuA9gKcrxlyvzX3Ow+l/T8TmaKK9G8N6R4SuPhP4gvNQvL5bmK5shJOmkxSPbFjL8sTGcFlbHJOzoOD2PsuXa34tL9RLex5zRWu2kWupa9Z6Z4SnvdSe7ZYoxdWiWzmRjgKFWWQY6clh344zWw3hnwompf2U/jCX7cG8o3A0zNiJOmPO83fszxv8AKx3xjmnZsDkKKuavpV3oesXel6lF5V1aStFKmc4YHse49D3FU6lNNXQ2rOwUV6F4n1CHwDer4Z0XSNIlntYYXvdQvtPiu5Z5njV2C+arKiDcFAUA8ZJOaxfGHiLRfEFlo50jQ7fR7uCCRb9baIJHLKXJDLySRjsfu9BwBTemwlqrnL0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBueJP8Al2/4F/SsOvUfjP4N0/wl/Yv9mzXMv2rz9/nspxt8vGMKP7xry6ta0lObkjKjFwgos9FbWrLT/h34YbRfFVlBruiXVxfCBYLjzA0hjKqpMPllhsOQW2+5p8fiqJvBPiDUNU8S2l94j1a+sr5bYQTB90LvuDHyljGAykBWxgEDsD5vRWfM/wAvwaf6I0suXl6a/jf/ADf3nd65beF/FfiGfX08V2+kLqUrXN1Y3lpcSTW8jHLqhjjZJFyTgll4xnFZ/iLxeLnxhp2p+H/Ngg0WG2t9PeYDzCsAG13A4ySCcDpnFcpRSWjVug3re/U9G8e+JfDUmgtZeC5WYa3fHVNTQxMht22jZbZIG5VZpTxkcjBq3Z+OpdV8HaVZW/jvUPCOo6Tbi08oS3QtbuMMSjjyAxRwDg5U5wOa8uooVkrf1povw0/4Idb/ANd/zOp8ZanNfQ2Mdz44vvFc0e4v53nmG3Jx9xpiGJOOfkXoOvblqKKB3Ov8P6tpd/4LvPCeu339lxterqFnfGFpEWUIUZJFQFtpXGCoJBHQ5rcsW8IaL8M/FGjw+LIL3WNUht3QpZ3CW/7qXf5Ss0e4ucdWVV5HPUjzSijzJsegeCI9B0i21WbUfF+kwvqeiz2aQfZ71nhkkAwHxblcDHJUt7ZrD0zSfDlv4usrXXfEcc+jurPcX2lRTbojtbaAJYlYncF6KRg9R25uih6u/wDX9ajW1j1rS/FNrp15M/iT4jSeKtC8h0OjzxXcj3OVwilZk8uMg4O4OSMcZrA+HPiODwjYeItVTW4rHVJtNe0sYFilaZpTJG4YEIUC4VhywOe3euEoo1V7dVb8/wDMOx2HiseHtZ0+HxDpup2Nnqtym7UNGjhnGJt2C8TeXsAI+YqWwvIBPAHH0UUaLYNepb0yzgv70QXWpWumxkEm4ullZB7YjR25+ldv46Gg6vp+iSab4u0qeTSdEhspIBb3ivNLGzk7C0AXB3DBYr74rz2ih6qwktbhT4UWSZEeVYVZgDI4JVAT1OATgewJ9qZRQM9I1uDw7qXgbwxo0HjfRVudI+1+ez21+EbzZQ67SLbJ4HOQPxrL+Gz6Ro/j3T9Y1jxBY2Vrpl6kh3w3LtcKCeYwkTf+PbTzXF0UR913RMoqUXFm1r2l2FoWubHxHpmqmWY/ubSK5V0Bydx82FFx24JPPSug1qXQfG8lrrFx4itdE1H7PDb3tre287qxiQRiSJoo3yGVQdrbcHPJFcLRQtFYuTcpOb3f6nrmieJ/DMEy6Smux2umaXod5p1vd3ltNuvJ7neWcJGj7UDEcNg455OQMu017Q4NB8AK+sQedoOryyXsIhm3CJ7hHEinZtK7UJxndyOOuPN6KFo7/wBbt/qTZWa/rZL9DW8V3dvqHjLWryxlE1tcX88sMoUqHRpGKtggEZBBwQDWZBGstxHHJMkCO4VpZAxVAT947QTgdeAT6A0yilFJKyKbu7npXiyPQNc1XSLrTvHekW32HS7Szd2t79XEkUYVmXFv0yOOQfpWH8Sde0/X/EVrJpd1LfrZ6fBZzajNGUe+kQEGYg885A+bnCjNcjRTerv5t/N/8OH+VgrsvCWoaRJ4M8R+HdX1RNJl1B7ae2upoJJIi0JfKN5aswzv4IU9DXG0UdGu/wDnf80I7G1u9E8C+JdE1fQda/4SC6tLgTXCJaPBCFGPkVpMOSRu52ADjrStoPgx9UN4PGaLpbHzfspsZ/toHXy8bPK3dt3mY7+1cbRTTaD+v6+82fF/iA+KfF+pa0YfIF5MXSLOdijhQT3OAMn1rGooqYpRVkNu7udzrF1ofjqSDV73XbfQtYMUcN/HeW8zxTlECCWNoUcglVXKkDnODWD4ltfDllLbW3hm/u9TaND9qvZovKilcnjyoyNwUDgljknsO+JRTEtFYKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA9w/aP8A+Zb/AO3r/wBpV4fXr/x48QaZrv8AYP8AZVz5/k/aPM/dsu3PlY+8B6GvIKcouLtJWFGSkrp3O60PwFpGo6b4am1PXr20ufEd1JbW0VvpiTpGVlWPLuZ0OCWB4U96j1jwRpFroGu6hpGuXt3Lod7FZ3EN1pqQK7OzrlGWd8geWeoHUVR8BXt1N8QfCdrNczSW9vq0HkwvISkW6ZS20dBkjJx1qp4ovLmLxR4gtIriZLafUpWlhVyEkKyPtLL0JGTjPTJqZ6cqXW/4OP6Njj1v/W5YTwrcWngu+1bWdC8QwO/kNp16LMrZMjN8xkdh3BXZtPJNZ2meGNf1u2a50bQ9S1CBJBE0tpaSSqr8fKSoIzyOPcV0Hhf/AJJZ46/3bD/0oq38P76607wN49uLC4ktphpsCCSJirANcIrDI9QSPxp7tv8ArYV9UvNL73b9TmZfB3iaCG7lm8OatHHZf8fTvYyhbfjd852/LwQeccGotJ8Ma/r8ckmhaHqWppEdsjWdpJMEPoSoOK9o0rVdQuPiN8NrGe9uJLV/DAZ4WkJVi0M4Ykd8hVHP90elc5rc2lD4Y+FZhpWuXujR2ey4fTdUS3givBI3meaht5MOcqQxbkYwOKHo35O35/5Ane1uqv8AkeW31hd6ZeSWmpWs9pcxHEkNxGY3Q+6nkVBXaeNtabWfDug50HV7SK3EqW+papdfaHuoyVIjEghjDKhJx1xvxwK4ugrzL2k6Lqeu3htdF0281KdV3tFZwNK4UEAnCgnHI596s+K9Ps9L8UXtlptpqtnaxMojg1iMR3SfKCfMUAAHJJHtisiu0+L3/JU9V/3bf/0njob2RJgaX4V8Q65bPcaLoOp6jAh2tLaWckqqfQlQQKp3+m32lXP2fU7K4s5yocRXETRttPQ4IBwfWu+1C/uLDwX4bsvGnhrVltIYPP0u807UxbRyxyMXDFTFIhkyx5G1sYzWb8U7a+g8TWUmpajdX7XOmW80JvowlzDEVISKbHVwBy3U5BPWiWj072/r+vmNar5HFV0nh/QtOl0HUPEHiF7n+zbORLeOC0dUlup3DEIHZWCAKrMWKnsAOa5uu40sHVPgprGm2Y8y707V4dSliUZYwGJomcD0ViuT23Cn0b/rcXVL+tjN1/QdNHhqx8SeG/ta6dcTvZ3FveSLJJbXCgNt3qqh1ZWBB2joQa5mu4vojpnwQ0+1vcxXOp6099bxMMM0CQ+X5mPQsSAe+DisjSPAfiXXvDN/4g0nS3udL07d9quBKi7Nqhj8pYM2AQeAanZvsv8AJfqNapef+ZX8K6B/wkeurZy3H2S1iikubu5K7vJhjUu7Y7nA4HckVsDQ/DviHRdWn8KQ6ra32kw/a3hv7mOdbi2DBXYFI02Mu5Tt+bjPPFSfC2VH8RanpjOqTaxo93p9sWOAZpI/kX6kjaPcipvAlrLpek+MdT1KN7a2h0efTyZVKk3MpVViwf4uGJHYA5py0Wna/wA9f+Bp5+aFG9/nb5af8H7vJmZpuiaPY+E4/EHicXs6Xlw9tY2VlOkDy7ApkkaRkcBRuUAbSST2xTPEnhmC0fSL3w81xPpmuRb7MXG3zUdX2PExGASrDqAAQQcCtPVQdV+C/h6WzBk/sS+u4L1VGTF5xR42Pop2sAfUYrQ1e5Tw3pHw5tdRBS7055NQurcj5oo5LhXQEdiVTOOuCPWqsnNJ91+X9MltqDa31/P+kQDwr4STxaPBs1xqZ1cyfY21RJ4/sq3ZO3Z5Pl7igf5S3mA9TjtWHoPhaO41DV28QyS2ljoUbSX/AJODIWDiNYkzxuZyBk8Dk84xXUXHh+8n/aIIiXdbz6v/AGot0P8AVm0aXzvO3dNmw5z04pNMvYfEa/EvT9MJluNXP22yjAO6dYrrzWVR1LbCWx1+U1Cb0b7O/lb+n93qW1ZtLuvx/rc56+0PRtS8H3Gv+F47+2/s6dIb+0vrhJyiyZ8uRZFRMglSpBXIJHPNPTRfD2haDpd54pj1K8utWQ3EVrYXMcHkW4YoHZnjfczFWwoA4HJ5q1okR034O+KLq9zCmq3Fpa2QYYM7RyGSQj1CgDJ6ZIFO8bxSax4c8Garp8bTwf2Umlv5altlxFI+Yzj+Ih1YDuDT2v6r8r/162Fq38n+f/D/AHepRv8AwLMPHmn6BpM/nwauIZtPuZV274JgCrsB0IGQ3upq9Z+HvCfiLULzQ/DLat/akcMj2V3czxtDfNGpZl8oRhotyqxXLt2B611A1Sz8PfGfwRBqM6RnR9NtLC/Yt8sEpRwwJ/2TIM+mD6Vg/DrRbzw78U5ptYhe1g8PR3E1/JIpAjURuq9f7zFQvrkYpP4X/wBvfh1+X4jV7/8AgP49Pn+BxGl6NqmuXRttF0271GcLuMVpA0rAeuFBOKl1Tw5reiIra1o+oacrPsBu7V4gWxnHzAc45xXVeEbnU9P+H2tyXGgXmo+G724jhu7nT7v7PLFKikgMwV/kw/IZCucc5qTxMr3fwn0m/wBOvNUh0WPUpIIdM1SRJmSTy8mSGVUTdHgAFdoCt9actFp5fiEdd/M8/ooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOp8a/8uX/AG0/9lrlq6nxr/y5f9tP/Za5auzG/wC8S+X5HJg/4Efn+Zo6R4j1vw+ZToOsahphmx5psrp4fMxnGdpGcZPX1o1bxDrWvyRSa7q9/qbwgiNry5eYoD1ALE4rOorjOs6QfEbxsqgL4x18AcADVJuP/Hqr/wDCb+K/9J/4qfWf9MObn/iYS/vztC/P83zfKAOewArDoo3DY1U8U+IIvsPl67qSf2aCLLbeSD7KCMER8/JkccY4qPSfEOtaA0jaFq9/pjSgCQ2dy8JfHTO0jNZ1FAFzU9Z1PW7kXGs6jd6hOBgS3c7SsB6ZYk1ToooA0dI8R634fMp0HWNQ0wzY802V08PmYzjO0jOMnr61a1Dxr4q1axkstV8Taxe2smN8FzfyyRvg5GVZiDyAaxKKNw2Nuz8a+KtOiWLT/EusWsaIsapBfyoFVRgKAG6AcAdqybq7ub66kur64lubiVt0k0zl3c+pJ5JqKigAq1p2qX+j3q3mkX1zY3SghZ7WZonAPXDKQaq0UAW9S1XUNZvDd6xf3V/csApnupmlcgdBuYk1Jaa7q9hptxp1jql7bWN1/wAfFrDcOkU3GPmQHDcetUKKPIBUdo5FeNijqQVZTgg+orS1XxNr2uxRRa5reo6lHCcxpeXckwQ+wYnFZlFAF7Std1bQZ3n0PVL3TZXXa8lncPCzD0JUjIqveXl1qF5Jd39zNdXMrbpJppC7ufUseSahooA1R4p8QDRv7IGu6kNM27fsQvJPJ2+mzO3H4Vn2t3c2N3HdWNxLbXETbo5oXKOh9QRyDUVFHW4dLF/Vte1fXpkl13Vb7UpY12o95cvMyj0BYnAqTSfEuu6CkiaHrWo6aspzItndPCH+u0jNZlFADpJHllaSV2d3JZmY5LE9STWne+KfEGo6YmnahrupXdjHjZaz3kjxLjphCcDH0rKoo6WDrc0tN8R65osYj0fWdQ09A5kC2t08Q3EAFsKRzgAZ9qj1XXdW16dJtc1S91KVF2pJeXDzMo9AWJwKo0UAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB1PjX/AJcv+2n/ALLXLV1PjX/ly/7af+y1y1dmN/3iXy/I5MH/AAI/P8wooorjOsKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA6nxr/y5f8AbT/2WuWrqfGv/Ll/20/9lrlq7Mb/ALxL5fkcmD/gR+f5hRRRXGdYUUV2vwpktpfG9tpl9plhfwX2Vf7ZbiUx7UZspngEkc8Gsq1T2VOVS17K5FSXJBy7HFUV7Hob6f4u8W6t4cvvCOkQWELTIL6wtPJkg2MQrFxxk49vy4ri9G+G2qeINDbUtIv9LumSPe1lHcE3CckBWULhScHGSBXJDHU9fae7a347fkZqvG7UtLHIUV1ms/DjWtGgt5DLZXzz3ItPJsp/MeOYgkIwwMHANX/+FR67uNv9v0c6iI/M/s77aPtGMdNuMfrj3rX67h0k+dFe2p9zhKK6XRPAWt65c3saxxWMdgxW6uL2Tyo4WHUE4PP4Vp3vwp1ay0C51p9W0WWwt4jKJorpmWXH8KHbgkngDjJIFOWLoRlyuav/AJg6sE7XOHort7P4U61qekx6hpF/pWoo7IrR2tyXaMtgkN8uFIByQT+dUdU8A32k6hp1vc6ppDRagZBFeJd/uFKfeDOQMYyB9TihYyhKXKpa/wBMFWpvZnLUV2U/w01IaZdXumatousC0TzJotOvPNkVe5xgfz+lWdM+EmsarpFjqVtq2jCC+VTCHuWBLEfc4TG8cggZwQfSk8bh0ruYOtTSu2cJRXXeIvhrrPhvSJdRuriwuYoJFjuEtZy7wM2MbwQMdR69RXI1tSrU60eam7ouMoyV4hRW14W8Pr4j1lLSTUbKwTKlmu5/L3gsBtTg5bngV6t4qtLDw5LdQaXY+ARbWcAIt9R+e9chM4xwSx7DPPHPNc1fGRpVFSSu2ZSrJT5EtTw+iu++Hnhe8v5PtunTeFr2eaN4xp2ruZHUAjL+WBkHjg+hNcpoWiS6/rcWmQ3dpaSzEqj3chRC3Zc4PJPAGOa2WIg5Tjf4d/x/yLVSOvkZlFd/L8Htbiu2szquiG92lo7T7YRLKAOqqV/nisPwx4dnvtfaKS80iyuLCdS1vrE2xJnDY8vGDu5GCPepji6E4uUZXsL20LNp7HOUV0HjqxutO8ZXttf2+nW1wnllotMQpbrlFI2g8jg8++a0LD4bandaTa6he6no+kxXihrZdRvPKaVT0IAB65+tP6zTVONSTsmU6kYpNvc4+ivUvCHht/C/xKtPDev2Ojasmpw+cZHi8/y1CSEbCwGCSvPByMVm6x8LdUa+1KXTrzSJ5YmeY6bbXObiNMkgbAMDA7Z+lYfX6Kqcrdla6ffoZ+3hzWfl+J5/RXSeH/Auq+INOl1JJbPT9OiO03moTeVET6A4OfrjFL4g8Baz4et7a5k+z39pdMEhurCTzY3Y9FHAOT9Oa6PrNHn9nzK5ftYX5bnNUV6ToPw01nS9asZ71/D73ZHmDSNRn3ySKQQcx4wSM5HOMgVmfFuwtNN8fTW+n2kFpCIIyIoIwigkcnA4rGGNpVKypQ1um7+hMa0Zz5Y9rnE0UV2Oq/DHWdM062u4bqw1M3Q3QW+nSPNLKuMl1XZyoyMkdMiumpWp02lN2vsXKcYtJvc46iup1PwDfaXoen6tPqWlyWt/OlujRTsfLYg535UABdpDcnBrS8TeHLzRvh7ZMzeGbyzS88tdQ0wmS4kYh22vJgAqOePZayeLpXiou93YlVYtpJ7nCUV3Pwq8O22teLbaa/n017eByHsbtwZLjKNjZGQQ2Dgn6U3xF4Sj8PaleatBrHhy7itbzemmJdeZIV8ziNo8DIHRhnoDUyxdONb2PW359A9qudw6nEUV6/4s8FxeI9G8O6pp/wDwj/h9bmxSSfzGFrHJI4U7VABJxz1rzPxD4ev/AAxq76dqsarMqhlZG3K6noynuKMPjKdfRaS109HYKdWNTbczKKK6Tw/4F1XxBp0upJLZ6fp0R2m81CbyoifQHBz9cYroqVIUo803ZFylGKvJnN0V1Os/D7VtE+xS3M9jLYXsqxR6hBcbrdSTj5mwMDvnGODWrP8AB7X4dSsLP7bpkjX4dopI5ZCgCgEknZ3B4xmsHjcOknzrW/4bkOtTSvc4GiunufA09l4pbQrvXNEhmSLzHuJLsrChzjYWK5D+2Kn+I+nXmm69apf2ui2zS2ayIuixlIWUs2GORyxx19MVSxNOU4xi731GqkXKyORor1HwF4MsdR8E69dXl5okstxaDyGlmBewYb/mk4/d5wDkdhXD+IPDn/CPm3H9saTqfnbv+Qbded5eMfe4GM54+hqaeLpzqypLdf5XFCrGcnFdDGoora8M+EtU8WXckOlRxhIV3TTzPsjiHqx/DtXROcacXKbskXKSirsxaK7Of4ZakmnX19Z6vomowWEDTXDWV55pQKM7SNvUgHHbg803T/hlrF3ptte3t5pekx3ePs6ajdeU82em0YP5da5/rmHtfnRHtqfc46iujm8A+IoPFMfh9rEteyjdHtYFGT+/u6bfr9OvFei/D74fSaXrlxDqg8M65bN8tzGrC4mtWAOMBlAXJOD16Cs6+PoUafPdPS9u5NSvCCve54vRXX2fgLU9cm1G+gey03TILiRPtd9L5MOQxGAcH6dMdqp+JPA2reGbOC9uWtbywnOI7yyl8yIn0zgHt6YraOKoykocyv2/T1LVSDlyp6nOUUV3Hwp0m0v/ABNc3uoQLcQaXaPdCJxkO4xtyO/c/UCrr1lRpSqPoOpPki5HD0V6rp+v3XxH8I+JbfxGkMs+nWzX1lNHEEMO0ElBjtwBzzgn2ryqooVpVHKM42cfO++u+goTcrprVBRRRXSaBRRRQB1PjX/ly/7af+y1V1bw/p+neGdN1O11+1vbq8x5tjEB5lvxn5vmPTpyB7Zq141/5cv+2n/stctXRmEZPEtp2tb56HFg0/Yxd+/5l/QtPt9V1y1sb3UItOgmba91N92MYJyeR6Y6jrRrmn2+la3dWVnfxajBC+1LqEfLKMdRyfp1PSqFFcnLLn5r6W2/XuddnzXub+leH9P1DwvqOqXOv2tldWmfKsJQPMuOAfl+YHnOOAenOK1vhTHbReN7bU77U7Cwgscs/wBsuBEZNyMuEzwSCeeRXFUVlUoyqQnBy+L00REoOUXFvc9N+IHiPxKlvcR/8JlpeoabdzuiW2mzxtIkZJIDlUBxjg8nPvVv4f2NvpHhLWy/irQ7S61izRbYf2iEkgbDff6FSNw6Z715PRXK8BH2HsYu3eyWtiXRvHlX5HaaPHb/AA+8eaRf3up6bqtupZpH0y488RgqU54HI3Zx7Vvf8I1oY8aHxIfHOljT/tP23if/AErO7ds2dc9s9favLaK0nhZTfNz2drPRaoJUXJ3vurM9hvfFOh+PNN1/Rv7Si0V7m9W6tZrs7IplVVXDntnbnn1HXGKbNo9toXwK1y0g1q01WQ3EUkrWcvmRREyRjaD34Ge30rzHQ9Y/sPUDd/2dYaj8hTydQg82PnHO3I5461q67481LXNHTSVtNP0zT1fzGttOt/JR29WGTXE8BOEo06XwJxetulvn0MvYyjJKOydzudJ0i3sfhbqfh9PGHh2K9v7lZldNTCoI8IGVjjPIUjGCOa5jwvovh/SvHy2XivUtNvrRYC8c1vcF7ZpeNqs+BxjOc8ZxXD0V1xwckprnfveVtdjRUXytc259B6FqGh2Ut6l5qHgexkuLd44v7KKodpxw8pIB7cY569qzNMj0vSfD3hOzm8TaDLLpGoNNceTqCsCrs+NvTpvGc46GvD6K5v7KWvvvXy7Jr8mzN4VNNX3/AMrHr3iW40638E+MoI9c0u7mv9UW8t47a7WRmRpEOAOpIA5xkDHWvIaKK78Lh1h4uN73d/wS/Q6KcORNXOg8G6Umpa/BJLqmm6clpJHMzX9x5QcBhkKccn2rrPiNoFrrHiPVfEFh4n8PSW7IsiW41AGZ9kYBAUAgkleBmvM6KU8PKVZVVK1lbb0/yF7N8/Pc9R+FmlWeia3a6/qPiTQYYJLdwbdr8LOhYYAZSAB+dZOm+EI9J8W6RczeKfDcsS3izO0OohgioQ5ydoxkDA964SioeFm6kp8/xK2y21t+ZHsn73vbnr0senn48JrQ8QaMbFv9K837au0AKE2E9N+ecZ6VhT+G4NS+IOqX8fibw9Dbxal56tNfhRKrMJPkOMHAOD6EEV59RUQwUoW5Z7R5duiD2L113Vj0v4maJa6t4h1PxFYeJdAmt2RGW2S/DTvtRVICgEE5HrXR6fd3VloFhp8fi3wZq1kkKDytXYK0K4+4AGOcdOcGvEaKl4BypRpOd1HbRdrBKjzJJvbyPZLD/hE4/i9Z6noupaTYWljblrzMwhhklZHTEOeD1GcYGPfNU/AyWGlfFTWr6817R0tow+2U3ihZvNO4bCcBsdG9D615PRSeX3i4ub1jy/jf/MTw90032X3Hs2g69pc3gxPDbXXhk6hpNxIg/tmPzLS4G5sSRvkdc9cEn05qfUPFVtoGg26Taj4UkVLyGZtO0G2Y7tsisSHD7QcDOSvOMV4lRUvLIOd3LS97aEvCxe70PVdV0HR9b8bf8JPb+NtJhsppkuSJp9tzHjB2hOvGMDoR6VifF+6s77x2brTr61voJbWMiS2mEgUjIKkjgHjOPQiuForejg3TnGbnflVlotv6RpCk4y5m76WCva/BnjvRtH+Gdte6hNBNqmll7aK3LqJmjd1PyqTkjG3kf3a8UorTFYWGKgoT6O/9eo6tKNVJPoesfFHUNK8Qapoeh6Bqunpa/vZ5JmuAII3kbPzsM4PDfTd71JP4fs5fhbbeG18W+GftkWoG6aQ6kPLK7WGAcZzyO1eR0VzxwHJThThPSLvt1u/8yVRsopPb/g/5nXeD5rTwp8ULF9RvraW2tZmR7q2fzIjuQqGDY5GWHP1rT1vwZpc2oapqU3jTQws80kttFBN5zOzMSqsB9zkgE8gda8+oreWHk6iqRnZ2s9Fr1KlTblzKVj074hWdle6P4StLPX9GuJLSCPT5jFehwjEAeYcdIxtOWPtxVT4vyWV1q2k3Wn6pYX6CxW2cWlwJSjISSWx0B38euD6V55RUUsH7OUHzX5b/AIijR5Wnfa/4hXr/AIT8TaXqXw8tNEeTw/DqGnuxEXiCHdBKCSQytkYb5uep68c5ryCitcTho4iKi3azuVUpqpbyPZfEF3Fd+DW0Aa/4LtGvJo8W+mRMsSkOGLGUOQvTklOeneug0XWtB8LaZpXhiPxFZ38FxHKLi/GpKv2YhQcLgnaCThRkfjXz1RXBLKozh7OU9Lt7Le1vwMHhU1Zv/hzs7/wNat4murLTvFuhSWwjE8d3dXwUMGYjYSARvGMkehB74HTfEHQ7PxFcWF3YeKvDgWx01Ld0k1EbnZCxO0AHOc8dK8morqeFqOUZe01j5I29nLmUubb/AIH+R33w2vtPGleJtGv9Rt9Pm1SzEdvLdPsj3DeDlu33h+tYviHwvYaDpcMqeJdO1O+kl2tbWDeaqJgkt5n1AGMd/aj/AIVx43/6E3xB/wCCuf8A+JqjqvhPxHoNqt1rmgapptu7iNZbyykhRmIJCgsAM4BOPY1qsO41nUjK17XWnRWGqbU3JPf/ACsZNegeBtS0u78F634V1HU00ie/kSWG7l4jbGMox7D5e/qfpXn9FXXoqtDlbts/mtUOpDnXoejWejWvhPQNca48b6Y4vLN4UtNKmWdrhsHaGypKrk4OMdfvCtDxRa6N8Rrix1mx8Uabpji3SG4s9Sm8pocZJ2/3uv0968porleDk5+15/e72Xa2xn7F83Nza/Ly/wAj2yHx14c0/WtO0NNSea1ttLk099XC/dkbbgj/AGRs69OR2Gap+AdA0jwh4r/tK98a6HLF5bxRJb3a5kBH8eeF6ZxzzjmvH6Ky/s1KEoQm0pKz213+7dkfVUouMXoz23TvEmk3/huXQhc+G/tum3s23+2o/MtbhTIxEkb5A5B68k+nNZvjHWIofh5c6WNW8Kbp5EYWOhWzEEh1JIcPhendeeleSUU45ZCM1JS636bjjh1GV79bhXYfDXxDZ6D4mkTVnMdhqFu1pM4/5Z7sYY+3H61x9FehWpRrU3TlszecFOLiz06GLR/AHhTXxH4g0/WL7VoTaW0djKH2xnILPj7pwc49sZNeY0UVFCh7Jyk3dvdihDlvrdsKKKK6DQKKKKAPQL7SLPXNd0vTtQ1P+zVnLpHN9nM26QlAq4BGMk9egqeb4Y6KuvSaDB40ibWF+VbWbT3jVmxkLv3Ecgjpmq91/wAjl4Y/6/0/9GR10fijW/B3h34lahqk9jq17rlu6ssbPGtsJNg2kEfN0x1B57Vy5xUrxxzhTb1jeyS3Vu62111PJw8qipJQfTy7s8kvrObTtQuLK6ULPbStFIoOcMpIIz9RUFevfCfxZqmqeKdWtbyVTY3ENxfvbKigeazqSQ2N38RGM1hW/jqfxVf2vhvUbSws9Bu7qCNIIotptkDg4RwQQSOCT6nGK51iqyqODh8KTbv38rb6bfidvtpJu8dt9Tz6ivoXVtYtfD3iGPS7fxLBpFpHhU0hfD0kiyLjn5x97PquK5z/AIkGi/Fx9QstCvptPax851TTJB9kkLY80RuoIXC9cYyxx0xWFPM3NX9m9rrf7tUl91yFiXb4en9dDx2iveLzV77xH4X15PDnimHWo1tXE1tfaaYXjQg7grgKN2M4yDWL4K8Q6doXgGCC6mvfDN3K7P8A2oNN85LsbjjDFGyACB+HWqWYzcG/Z6ppW1vr/wBu3+5Mf1h8t1HW9v60PIaK9L+LX9q3Nnod7e31pqNg8cgtrqG2aCRydpO9CT6DGMd+BXmld+Gre3pKpa3/AA/ovyOinPngpdztvC/gXSPEPhu61WfxP9haxXfew/2e8nkKS207gw3ZC54BxU7+ANBk8OXGt6b4w+12drPFDcP/AGZInlh3VS2C2TgNnAHOMVe+F09tbeD/ABpNf2n222S1hMtv5pj8xf3nG4cj61L4zmk8by6rqXhPUcaNp1hA15bs8kQkYFyBsxhiuOp9ODXl1K1dYmUOdqKa1srK9tNr63sv1OXnn7Vxvpfy8tNvMz/EnwwTw2umXk+t+bo94wWW/Fm3+j5GVJQMSQfwq9H8IrOe/sYrbxP5lveafJqCz/2eRiNTHj5S+eRIT6jb05rsNV1yz/tvTPCmv4OlazosCAnH7qbLBWHpnA57ELXPfEu71PwXH4ctNNvfKmj0lrGWVEB3oNgOMg4ztB45HrXLSxWLqOFPmtJ3torNWeu26a/Iyp1as+VX1av/AFp/VihF8KdEm/sry/Gef7X3fYv+JU/73b1/j+Xr/FioZfhjoNvpV7qNz408u2sbprO5f+ypD5coIBXAfJ6jkZHvT/FGpXekeBfh/f6dMYbqCKdo5AoO05Tscg1q+DNbtIPhnqOr+I7f7fG/iBJp/nKYc+UfMwo5weduMHpWkquLVNVFUbXNa1o3+Jrt2/ErnrckZ339O78vQ5u98A6Fa6LYazH4v83S7q6Ns1z/AGY6+WQjnO3duPKgdO+e1XNS+GXh/SdZj0m/8bpBfShSkcmmOFO7pl9+0Z9zVfxvHeeI49Z8TaVqPneHY9QWNIGkkXMhjQFxGRgfexng13/iLVvCdr8VbO18Q6Kj3TQReVqMkzMiMSdoaL7uM/xc9fxoliMSlF87baldJRurJOzulqr6/kEqlRLR9H26fLz1PFPEegXfhjXrjStQ2mWEjDp911IyGHtisuup+JEOsQ+O7/8A4SB0kuWIZJI1KxtHj5SoPQYGOp5B5PWuWr28PN1KMZyd20tjui7xTOg8PeFW1/Q9c1FbsQf2RbrOYzHu83O7jOePun1rn69b+G3inWx4E8SxRXZI0myD2KiFD5THzCT0+bkDrmsjQ7vW/Hvi7Qf+E1M11pQneOOZ7ZYo2YoW2blUAlig4+tcSxVWNWrzpcsfPXa+mmt/X7zn9tKLm5bL/K553RXsI8Y+Kj8RG8Of2PAdL8/7P/Zf2Ndnkbsb846bec52+1ef+PdNstH8d6rY6WALWKUbFByEyoJX8CSPwrahi3VmoSja6urO+nnou5pCq5S5WtdznqKfEqvMiyNsRmAZsfdGetfQOtXNh4O+wafpPiOHw5Z+WpWJdEe6+1H1Mo4Yn259+aMVi/q8oxUbuV+/T0Tf4Cq1uSSile5890V7F4j1P+yPHWiat4HspJNR1GNobmE6dJbx3XKnKiRRjPJLAnGBk+vba7dXelabrep+GIY7vXpEhN5aCYSG2wmMhQMsQOx69fauKeaSioP2fxd3azvbXTReZk8S00uXfzPnPSbayvNUhg1XUP7OtHz5l15LS+XwSPkXk5OB+NQ3kUEN9PFaXH2mBJGWKfYU8xQeG2nkZHOD0ruvht4u1yPxxbWSXuLfVL4y3ieSn71mzk525H4YrnPGMjRfEHW5Izh01KZlOOhEhrvjVqPEOnJdL7+fon+PyN4yk5uL6Jfr5eQ+98KtZ+BNN8S/aw631w8H2fy8FNpYZ3Z5ztPYVz9ez6l478TwfCHRNattQI1C5vJIpphbxncoLgDbtwPujoK8s8RXGs3urtfeI451vbpFkLTQeUZFxtDBcAY+XGQO1Z4OtWqOSqW0bW+uj2tZaef4E0Zylfm7v8zLoor1eXULzwN8M9DvfCNqizalG0l7qfkiRkYYwmSCB1I5/unvmt8RXdLlUVdydlrZd9yqlRxaSWrPKKK9n0m/k8c6N4bg8U6bLsk1cm4v3ijijumEMhRcAgknCgkLj5cZzUI8Y+Kj8RG8Of2PAdL8/wCz/wBl/Y12eRuxvzjpt5znb7Vx/X58zhyK6u372llpo7f5GP1h66bb6+b/AMjx6ivd9KvT4afx/p+kNE9hpKLcWkEiBkjkZCzD1IBAGCeMVzVzrd54y+Dutah4hMV1eafdxC2nEKoyBmQEfKBxhiKUcwlJ35PdvFXvr71raW89dRqu3K1tL2+/U8tor23RvE+naV4S0nTxf3vg+8WJN/maT5i3bYHz5KHIPXOR1qZ9Em1X4v2z+KlsdRVdKM9iIYjGlyVbgOjE8/MxxnHA9xUvM3GUueFkr/O3bS34k/Wt7o8Mrp/B3hjSfE9wtneeIf7NvpZRHb2/2J5vN4zncCAPxruvCvivxL4r8XyaB4j0qG50yQslzaPaBFswAcEHGRzjqSfSuR8JW0Fl8ZLO1s38y3g1N44nzncoZgDn6CtJYmpKM4P3ZJc2jT/Nfp6Mc6suSXRpX7nM61p39j69f6b5vnfY7h4PM27d+1iM4ycZx0zVKuu1GztdQ+Ml3Z6g+y1n1p45TnHymUgjPbPTNer6trFr4e8Qx6Xb+JYNItI8KmkL4ekkWRcc/OPvZ9VxSqY6VKMI8vNJq/X9E/ysXUrckuVK+lz56orsPifaaVbeMS+hW8ttb3ECzNFJayW4VySDtR1U4OAeBjJNcxe6bfad5f8AaFlcWnmrvj8+Jk3r6jI5HvXfRrKrTjPa/R7m0ZKST7n6NV4l+1X/AMks03/sNRf+iJ69trxL9qv/AJJZpv8A2Gov/RE9bFHyTRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAd3rOv6l4b1GyvtFufs1zslTf5av8AKduRhgR2rI1r4g+J/EOmvp+san9otXYM0f2eJMkHI5VQal8a/wDLl/20/wDZa5atsfQpSxTnKKbVtba7HDg4R9jGVtf+CdLo3xC8UeH9Mj0/SNT+z2sZJWP7PE2CTk8spPU1W1/xpr/iiCGHXb/7VHCxaMeTGm0kY/hUVh0VxLD0VP2igr97K/3nWqcE7panWWnxR8ZWNoltb63J5ca7V8yGORsf7zKSfzqvbfEHxTaatPqcOsSm7uEEckkiI+VByFAYEAAk9AOprm6KX1XD6+4tfJC9lT25V9x02q/EbxZrdg9lqOsSSW8gw6JGke4ehKKCR7dKk034l+LdJ02Cw0/VjFbW67I0NvE20emSpNcrRR9Vw/Ly8it6IPZwtayNPXPEer+JLpbjW76S7kQYTdgKg74UAAfgKzKKK2jCMI8sVZFpJKyOl0f4ieKdA0uPTtJ1T7PaRElI/s8TYySTyyk9SatT/FXxncwmKbWdyEgkfZYR0OR/B6iuQorB4XDylzOmr+iI9lTf2V9x0mtfEHxP4h042Gr6n9otmYOU8iNORyDlVBqnq3ivW9dns5tWvjcyWQxAxjQFOQewGeg65rHoq44ejC3LBK3kuo/ZwXQ6jU/iT4s1nTJtP1LVfOtZ12yR/Zol3DOeoUHtUlj8T/F+m6fBZWWr+Vb28axxJ9mhO1QMAZKZPHrXJ0VH1XD8vL7NW9EL2cLW5UdhJ8VvGkpjMms5Mbb1/wBFh4OCP7nuaztf8b+IfE9nHa65qH2qGKTzEXyI0w2CM5VQehNYFFOOFw8GpRgk15IapwTuki/q2t6jrktvJqty1zJbQLbxMygERqSQCQOep5OTVCiit4xUVaKsikklZGnoXiPVvDV411ol69pK67WKgMGHurAg/lVrW/G3iPxEIRq+qyzrA4kjVVWMK46NhABkZPPXmsKis3RpOfO4q/e2pPJG97anWf8AC0PGf2L7L/bs3l7dufLTfj/f27s++c1yryPLI0krM7sSzMxyST1JNNoop0aVO/JFK/ZWHGEY/CrBXTaT8RvFeiaellpusSR28YwiPEkm0egLqSB7VzNFOpSp1VapFNeauEoxl8SudUnxM8Xx6g16NalM7J5eXijZQuc4ClcD8BzgUlv8S/Ftrf3V7Bq224vCpnf7NEd+0YXgrgcelctRWX1TD/8APtfcifZU/wCVfcdVY/EvxbpscyWWreWs073Eg+zRHdI7bmPK9yenSoNa+IHibxFprWGsan9otWYMY/s8SZI6cqoNc5RTWFoKXMoK/eyBU4J3SR0GheOvEnhqza00XVHt7dm3eWY0kAPtuBx+FZ2sa5qXiC/N7rN3Jd3G0IHfHCjJwAOAOTwPWqFFWqNKM+dRV+9tSlCKd0tQre0Lxv4j8NW7QaLqklvCxyYiiyKD6gMCB+FYNFVOnCouWauvMJRjJWkrm3rfjPxD4ieFtX1SafyG3xKoEao3ZgFAGffrWj/wtDxn9i+y/wBuzeXt258tN+P9/buz75zXJ0Vk8NQaUXBWXkhezh2R0mj/ABB8T6DZNaaVqjQwvI0rBoY5CzN1JLKTz9aZB488SW2gy6NDqW3T5kkR4fIjORISXG7bnnce/GeK56im8NQbu4L7kHs4dkdhD8V/GkECQxa0QkahFBtYTgAYHJTNYN34j1i+1pdXutRuH1BSClxvwyY7LjoOTwOOazaKI4ejB3jBL5IFTgtkjqrz4meML+xazudbm8lxtby40jYj/eVQ361BovxA8TeHdNWw0fU/s9qrFhH9nifBPXllJrnKKX1Why8vIreiF7OFrcqOj1r4geJvEWmtYaxqf2i1Zgxj+zxJkjpyqg1ZtPij4ysbRLa31uTy412r5kMcjY/3mUk/nXJ0UfVcPy8vIreiD2cLWsi7q2s6jrt+b3V7uS7uCAu+Q9AOwHQDk8D1qxrnifWPEn2b+271rr7KhSHKKu0HGfugZPA5PPFZVFaKlTVrRWm3l6FcsdNNj9I68S/ar/5JZpv/AGGov/RE9eJf8NB/E7/oZv8AyQtv/jdYfiv4peMfG+lR6b4n1j7daRTCdI/ssMeHCsoOUQHozcZxzWhRyVFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB1PjX/ly/7af+y1y1dT41/wCXL/tp/wCy1y1dmN/3iXy/I5MH/Aj8/wAwooorjOsKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA6nxr/AMuX/bT/ANlrlq6nxr/y5f8AbT/2WuWrsxv+8S+X5HJg/wCBH5/mFFFFcZ1hRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHU+Nf+XL/ALaf+y1y1dT41/5cv+2n/stctXZjf94l8vyOTB/wI/P8wooorjOsKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA6nxr/wAuX/bT/wBlrlq6nxr/AMuX/bT/ANlrlq7Mb/vEvl+RyYP+BH5/mFFFFcZ1hRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHU+Nf+XL/tp/7LXLV1PjX/ly/wC2n/stctXZjf8AeJfL8jkwf8CPz/MKKKK4zrCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAP/2Q==)

**GCD**

#include<stdio.h>

#include<time.h>

void delay()

{

int i;

for(i=8000;i>0;i--);

}

int gcd(int a,int b)

{

if(b!=0)

return gcd(b,a%b);

else return a;

}

void main()

{

int m,n,ans;

time\_t start,end;

start=time(NULL);

printf("enter two numbers");

scanf("%d %d", &m,&n);

delay();

ans=gcd(m,n);

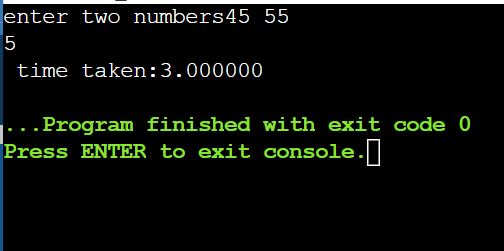
printf("%d",ans);

end=time(NULL);

printf("\n time taken:%f", difftime(end,start));

}

OUTPUT:



**PROGRAM-2** Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

**LINEAR SEARCH:**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int RecursiveLS(int arr[], int value, int index, int n)

{

int pos = 0;

if(index >= n)

{

return 0;

}

else if (arr[index] == value)

{

pos = index + 1;

return pos;

}

else

{

return RecursiveLS(arr, value, index+1, n);

}

return pos;

}

int main()

{

int n, value, pos, m = 0, arr[1000000],i;

time\_t start,end;

printf("Enter the total elements in the array ");

scanf("%d", &n);

for (i = 0; i < n; i++)

{

arr[i]=rand();

}

start=time(NULL);

printf("Array elements are\n");

for(i=0;i<n;i++)

{

printf("%d ",arr[i]);

}

printf("Enter the element to search ");

scanf("%d", &value);

pos = RecursiveLS(arr, value, 0, n);

if (pos != 0)

{

printf("Element found at pos %d ", pos);

}

else

{

printf("Element not found");

}

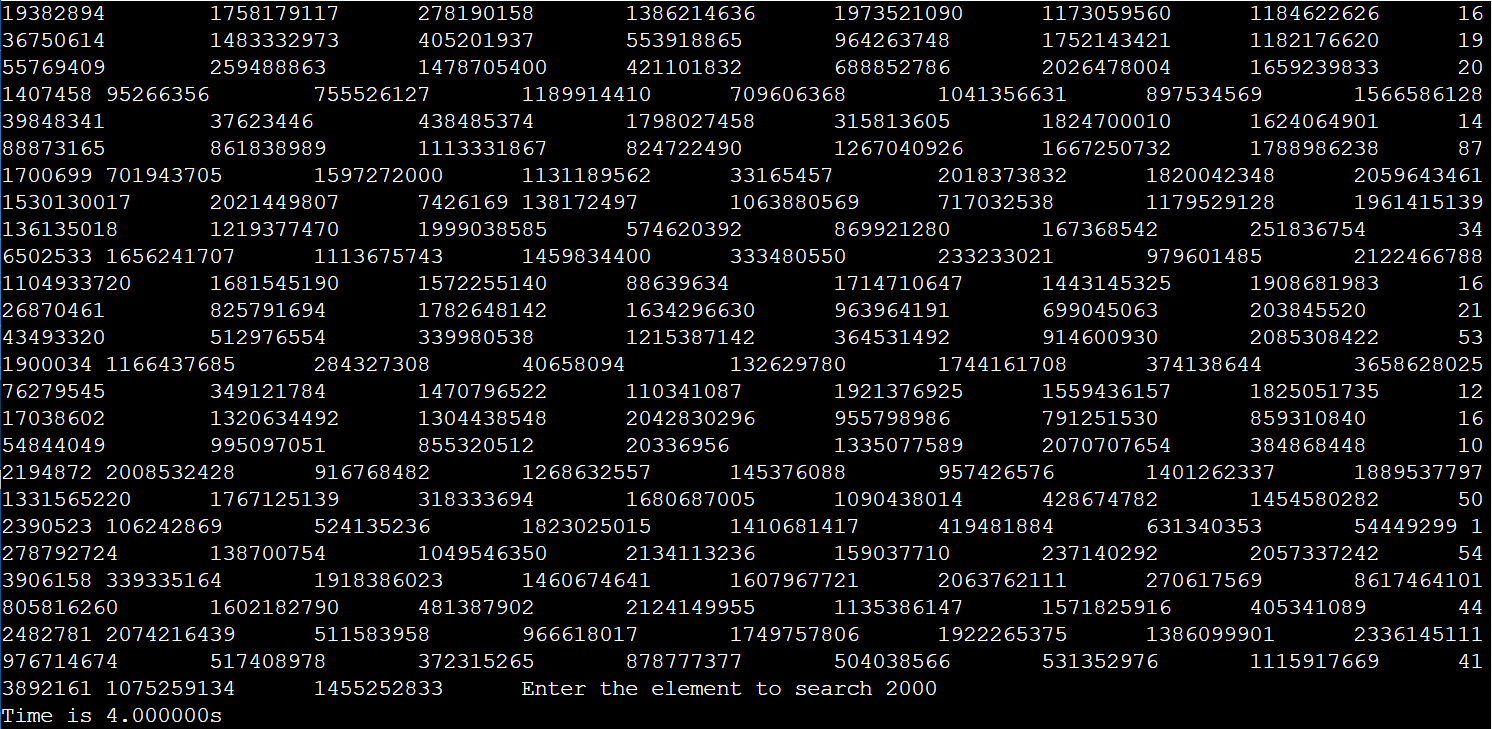
end=time(NULL);

printf("Time is %f",difftime(end,start));

return 0;

}

**OUTPUT:**



**BINARY SEARCH:**

#include <stdio.h>

#include <time.h>

void delay()

{

int u;

for(u=8000;u>0;u--); }

int recursiveBinarySearch(int array[], int start\_index, int end\_index, int element){

if (end\_index >= start\_index){

int middle = start\_index + (end\_index - start\_index )/2;

if (array[middle] == element)

return middle;

if (array[middle] > element)

return recursiveBinarySearch(array, start\_index, middle-1, element);

return recursiveBinarySearch(array, middle+1, end\_index, element);

} return -1; }

void main{

int array[10],n,element,i;

time\_t start,end;

printf("Enter the number of elements\n");

scanf("%d",&n);

start=time(NULL);

delay();

printf("Enter %d elements\n",n);

for(i=0;i<n;i++){

scanf("%d",&array[i]); }

int found\_index = recursiveBinarySearch(array, 0, n-1, element);

if(found\_index == -1 ) {

printf("Element not found in the array ");

}

else {

printf("Element found at index : %d",found\_index);

}

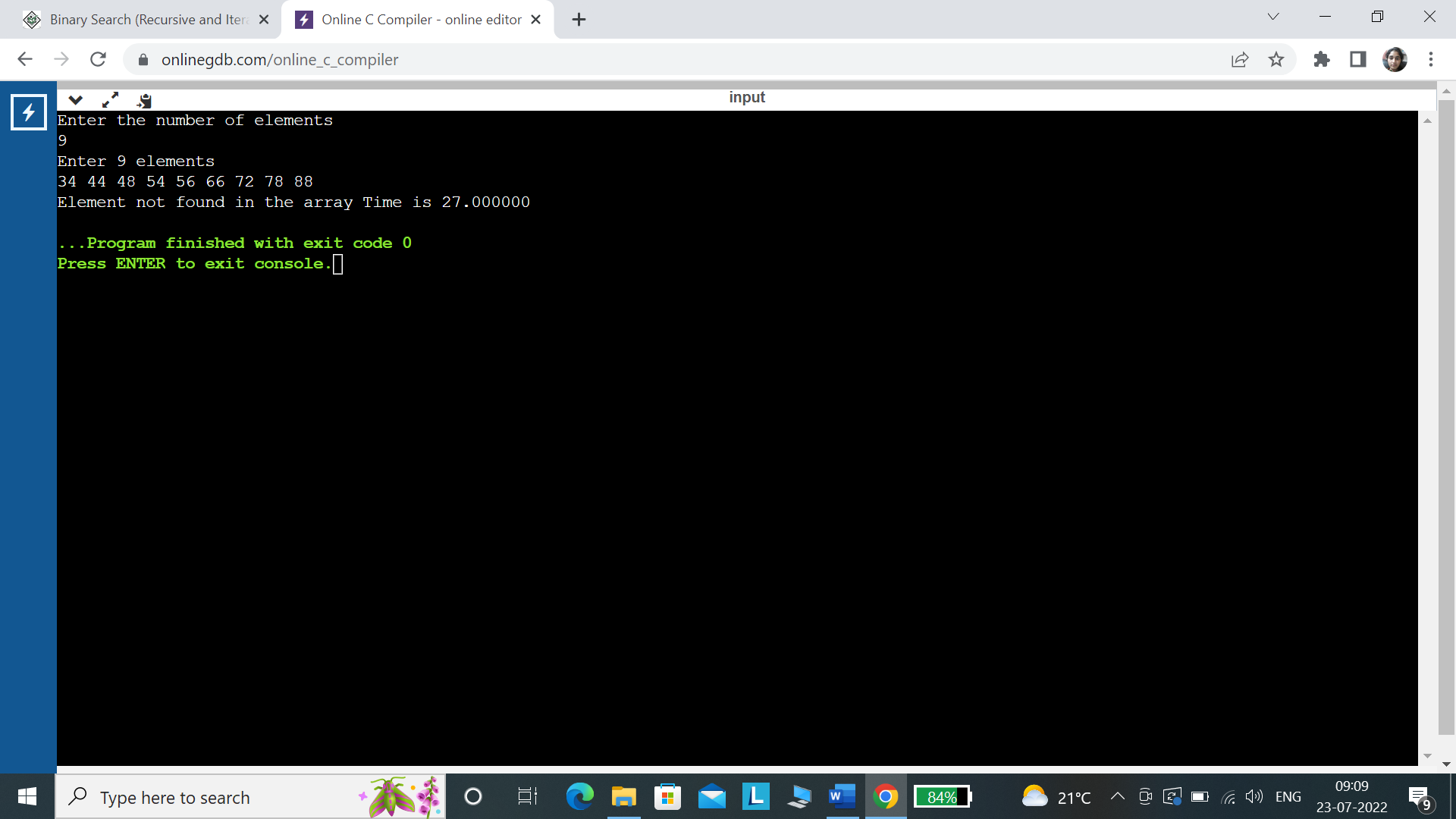
end=time(NULL);

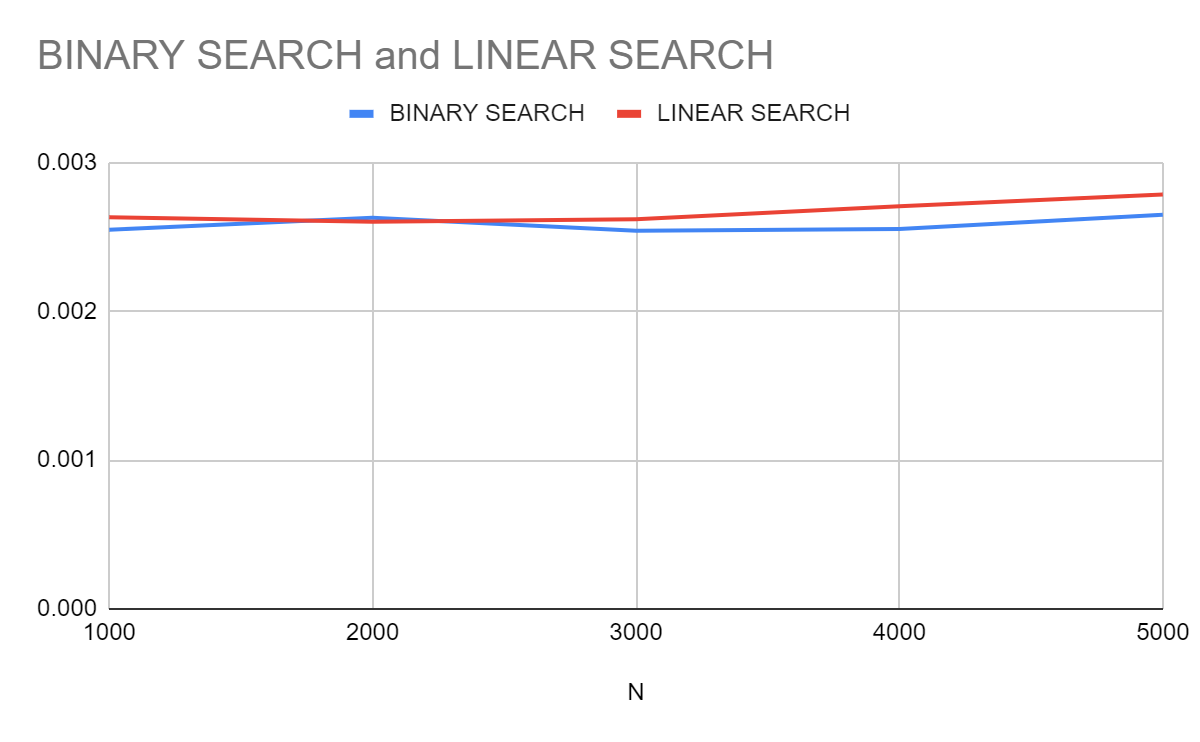
printf("Time is %f",difftime(end,start));

return 0;

}

**OUTPUT:**



**PROGRAM 3** Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

#include <stdio.h>

#include<time.h>

void main()

{

int a[100], n, i, j, position, swap;

time\_t start,end;

printf("Enter number of elements");

scanf("%d", &n);

for (i = 0; i < n; i++) {

a[i]=rand(); }

start=time(NULL);

for(i = 0; i < n - 1; i++)

{

position=i;

for(j = i + 1; j < n; j++)

{

if(a[position] > a[j])

position=j;

}

if(position != i)

{

swap=a[i];

a[i]=a[position];

a[position]=swap; } }

end=time(NULL);

printf("Sorted Array:");

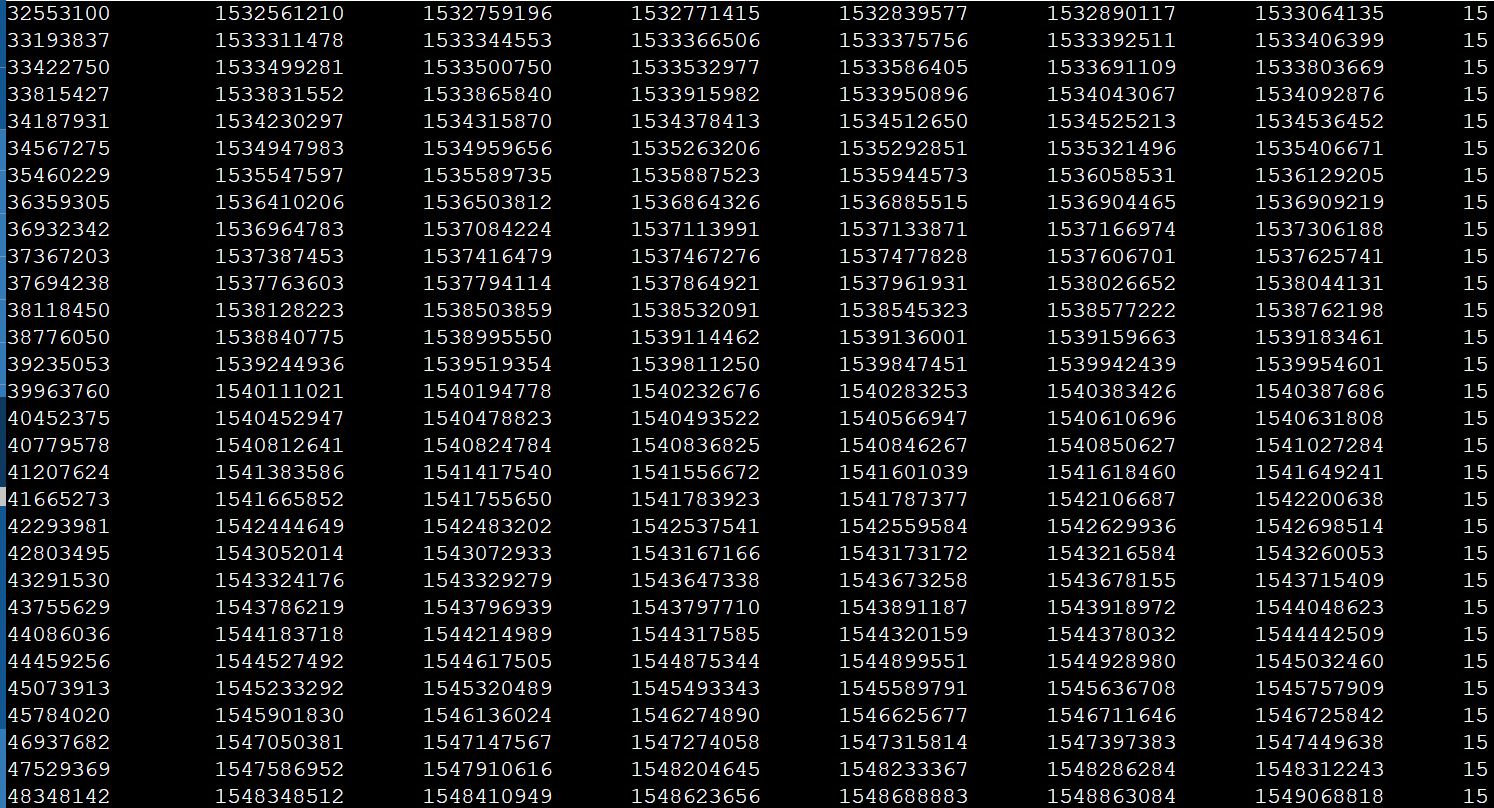
for(i = 0; i < n; i++)

printf("%d", a[i]);

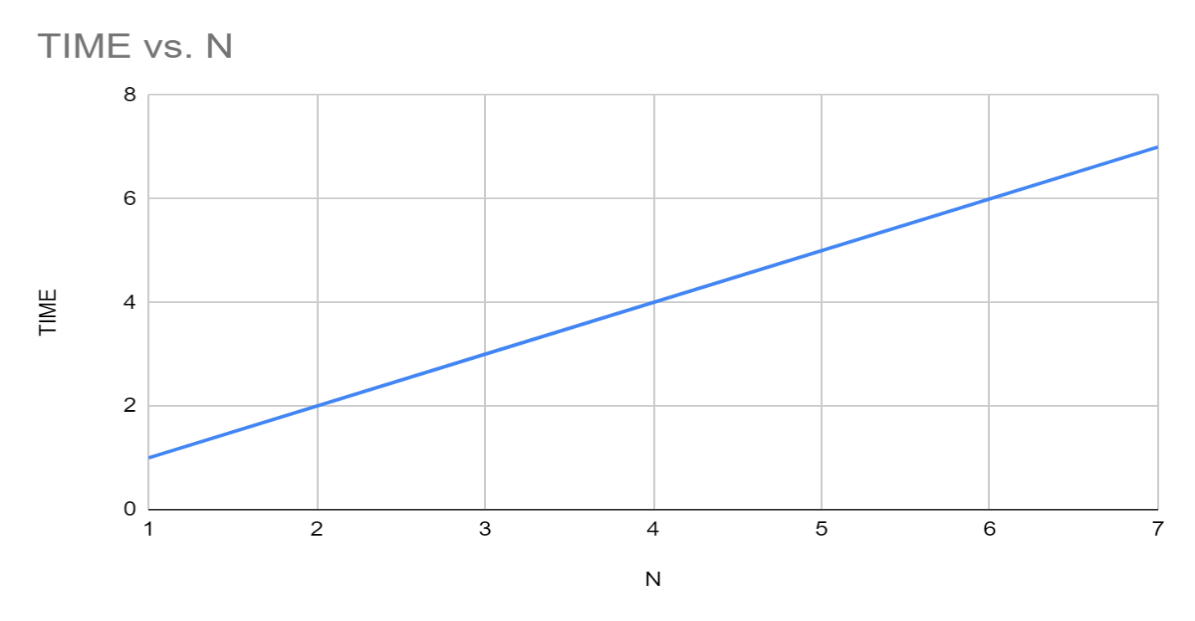
printf("Time is %fs",difftime(end,start));

}

**OUTPUT**



**GRAPH**



**PROGRAM 4 :** Write program to do the following:

**a)** Print all the nodes reachable from a given starting node in a digraph using BFS method.

**b)** Check whether a given graph is connected or not using DFS method.

**A:** #include<stdio.h>

#include<conio.h>

int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;

void bfs(int v)

{

for(i=1;i<=n;i++)

if(a[v][i] && !visited[i])

q[++r]=i;

if(f<=r)

{

visited[q[f]]=1;

bfs(q[f++]);

}

}

void main()

{

int v;

printf("\n Enter the number of vertices:");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

q[i]=0;

visited[i]=0;

}

printf("\n Enter graph data in matrix form:\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&a[i][j]);

printf("\n Enter the starting vertex:");

scanf("%d",&v);

bfs(v);

printf("\n The node which are reachable are:\n");

for(i=1;i<=n;i++)

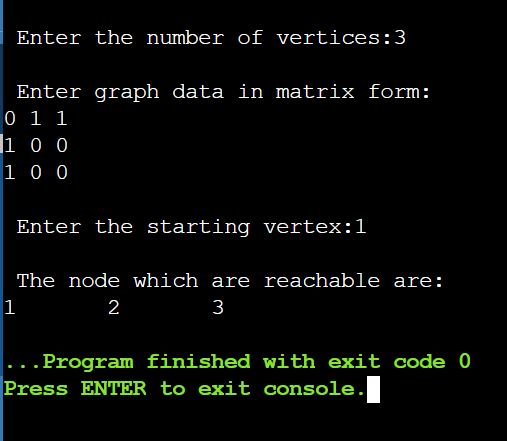
if(visited[i])

printf("%d\t",i);

getch();

}

**OUTPUT**



**B.** #include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<time.h>

int a[20][20],reach[20],n;

time\_t start,end;

void dfs(int v) {

int i;

reach[v]=1;

for(i=1;i<=n;i++)

if(a[v][i] && !reach[i]) {

printf("\n %d->%d",v,i);

dfs(i); } }

void main() {

int i,j,count=0;

printf("\n Enter number of vertices:");

scanf("%d",&n);

for(i=1;i<=n;i++) {

reach[i]=0;

for(j=1;j<=n;j++)

a[i][j]=0;

}

printf("\n Enter the adjacency matrix:\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&a[i][j]);

start=time(NULL);

dfs(1);

end=time(NULL);

printf("\n");

for(i=1;i<=n;i++)

{

if(reach[i])

count++;

}

if(count==n)

printf("\n Graph is connected");

else

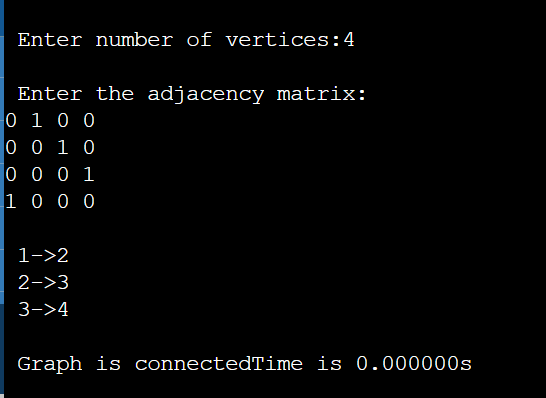
printf("\n Graph is not connected");

printf("Time is %fs",difftime(end,start));

getch();

}

**OUTPUT**



**PROGRAM 5** Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

#include <stdio.h>

#include<time.h>

void main()

{

int n, i, j, temp;

int arr[64];

time\_t start,end;

printf("Enter number of elements\n");

scanf("%d", &n);

for (i = 0; i < n; i++) {

arr[i]=rand(); }

start=time(NULL);

for (i = 1 ; i <= n - 1; i++)

{ j = i;

while ( j > 0 && arr[j-1] > arr[j])

{

temp = arr[j];

arr[j] = arr[j-1];

arr[j-1] = temp;

j--;

}

}

end=time(NULL);

printf("Sorted list in ascending order:\n");

for (i = 0; i <= n - 1; i++)

{

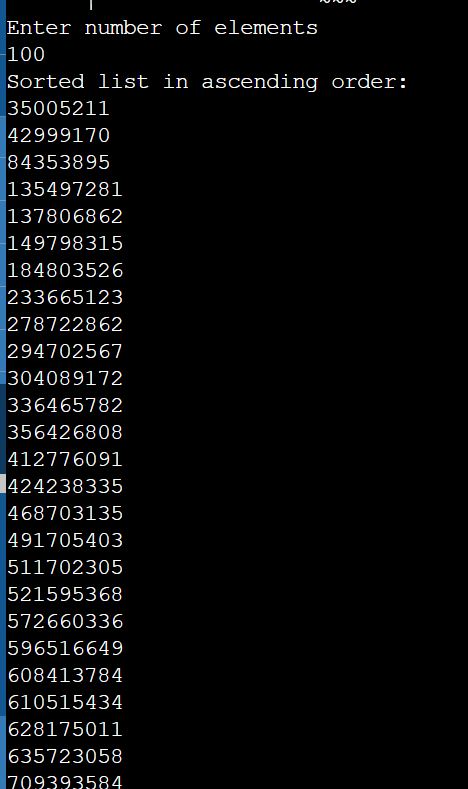
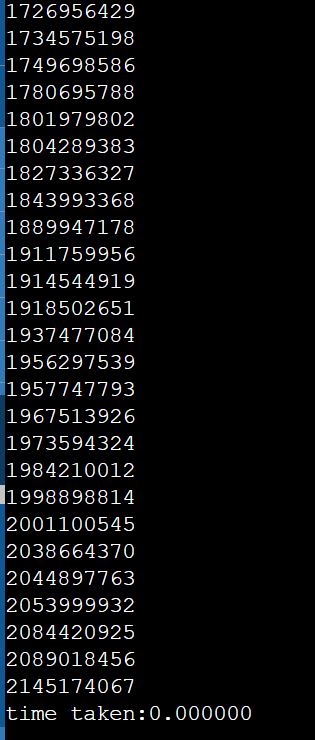
printf("%d\n", arr[i]);

}

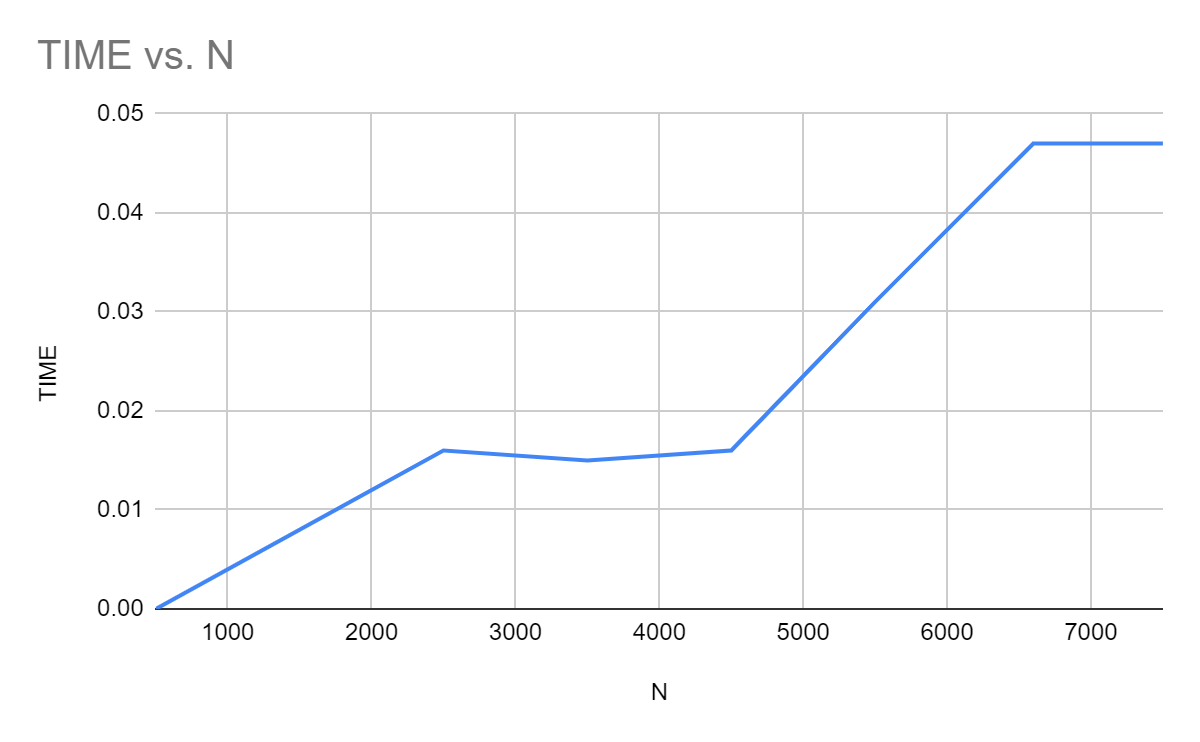
printf("time taken:%f",difftime(end,start));

}

**OUTPUT**

**GRAPH**

****

**PROGRAM 6** Write program to obtain the Topological ordering of vertices in a given digraph.

#include<stdio.h>

#include<conio.h>

void main()

{

int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;

printf("Enter the no of vertices:\n");

scanf("%d",&n);

printf("Enter the adjacency matrix:\n");

for(i=0;i<n;i++)

{

printf("Enter row %d\n",i+1);

for(j=0;j<n;j++)

scanf("%d",&a[i][j]);

}

for(i=0;i<n;i++)

{

indeg[i]=0;

flag[i]=0; }

for(i=0;i<n;i++)

for(j=0;j<n;j++)

indeg[i]=indeg[i]+a[j][i];

printf("\nThe topological order is:");

while(count<n)

{

for(k=0;k<n;k++){

if((indeg[k]==0) && (flag[k]==0)){

printf("%d ",(k+1));

flag [k]=1;

}

for(i=0;i<n;i++)

{

if(a[i][k]==1)

indeg[k]--;

}

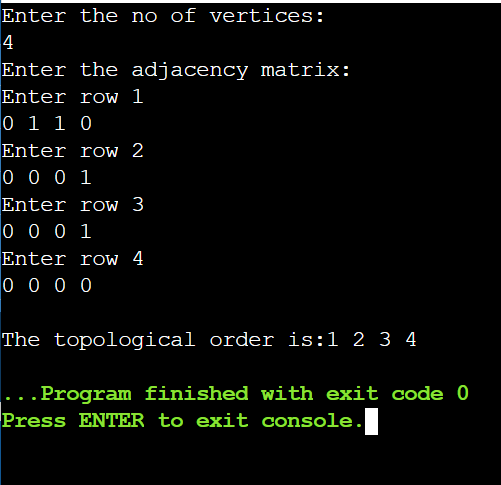
}

count++;

}

}

**OUTPUT**



**PROGRAM 7:** Implement Johnson Trotter algorithm to generate permutations.

#include <stdio.h>

int fact(int n) {

int f=1;

for(int i=1;i<=n;i++) {

f=f\*i; }

return f; }

int search(int a[],int mobile,int n) {

for(int i=0;i<n;i++) {

if(a[i]==mobile) {

return i; } }

return -1; }

int getMobile(int a[],int dir[],int n)

{ int mobile=0;

for(int i=0;i<n;i++) {

if(dir[a[i]-1]==0 && i!=0) {

if(a[i]>a[i-1] && a[i]>mobile) {

mobile=a[i]; } }

else if(dir[a[i]-1]==1 && i!=n-1) {

if(a[i]>a[i+1] && a[i]>mobile) {

mobile=a[i]; } } }

return mobile;

}

void Permutations(int a[],int dir[],int n) {

int mobile=getMobile(a,dir,n);

int pos=search(a,mobile,n);

if(dir[a[pos]-1]==0 ) {

int temp=a[pos];

a[pos]=a[pos-1];

a[pos-1]=temp; }

else if(dir[a[pos]-1]==1 ) {

int temp=a[pos];

a[pos]=a[pos+1];

a[pos+1]=temp; }

for(int i=0;i<n;i++) {

if(a[i]>mobile) {

if(dir[a[i]-1]==0) {

dir[a[i]-1]=1; }

else if(dir[a[i]-1]==1 ){

dir[a[i]-1]=0; } }

for(int i=0;i<n;i++) {

printf("%d\t",a[i]); }

printf("\n"); }

int main()

{

int n=4; int a[]={1,2,3,4};

for(int i=0;i<n;i++)

{

printf("%d\t",a[i]);

}

printf("\n");

int dir[]={0,0,0,0};

int total=fact(n);

int count=1;

// printf("%d",total);

for(int i=1;i<total;i++)

{

Permutations(a,dir,n);

count++;

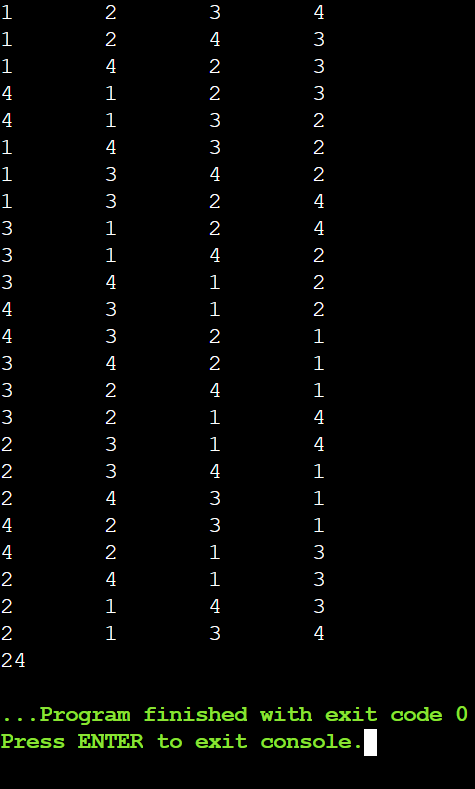
}

printf("%d",count);

return 0;

}

**OUTPUT**



**PROGRAM 8:** Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

#include <stdio.h>

#include <stdlib.h>

 #include<time.h>

void merge(int arr[], int l, int m, int r)

{

    int i, j, k;

    int n1 = m - l + 1;

    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)

        L[i] = arr[l + i];

    for (j = 0; j < n2; j++)

        R[j] = arr[m + 1 + j];

    i = 0;

    j = 0;

    k = l;

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) {

            arr[k] = L[i];

            i++; }

        else {

            arr[k] = R[j];

            j++; }

        k++; }

    while (i < n1) {

        arr[k] = L[i];

        i++;

        k++; }

    while (j < n2) {

        arr[k] = R[j];

        j++;

        k++; } }

void mergeSort(int arr[], int l, int r)

{

    if (l < r) {

int m = l + (r - l) / 2;

        mergeSort(arr, l, m);

        mergeSort(arr, m + 1, r);

  merge(arr, l, m, r);   } }

int main()

{

    int i,n;

    int arr[1000];

      time\_t start, end;

    printf("enter the number of elements");

    scanf("%d",&n);

    for(i=0;i<=n;i++)

    {

        arr[i]=rand();

    }

    start=time(NULL);

    int arr\_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is \n");

    for (i = 0; i < n ;i++)

        printf("%d ", arr[i]);

    printf("\n");

    mergeSort(arr, 0, arr\_size - 1);

    for (i = 0; i < n; i++)

        printf("%d ", arr [i]);

    printf("\n");

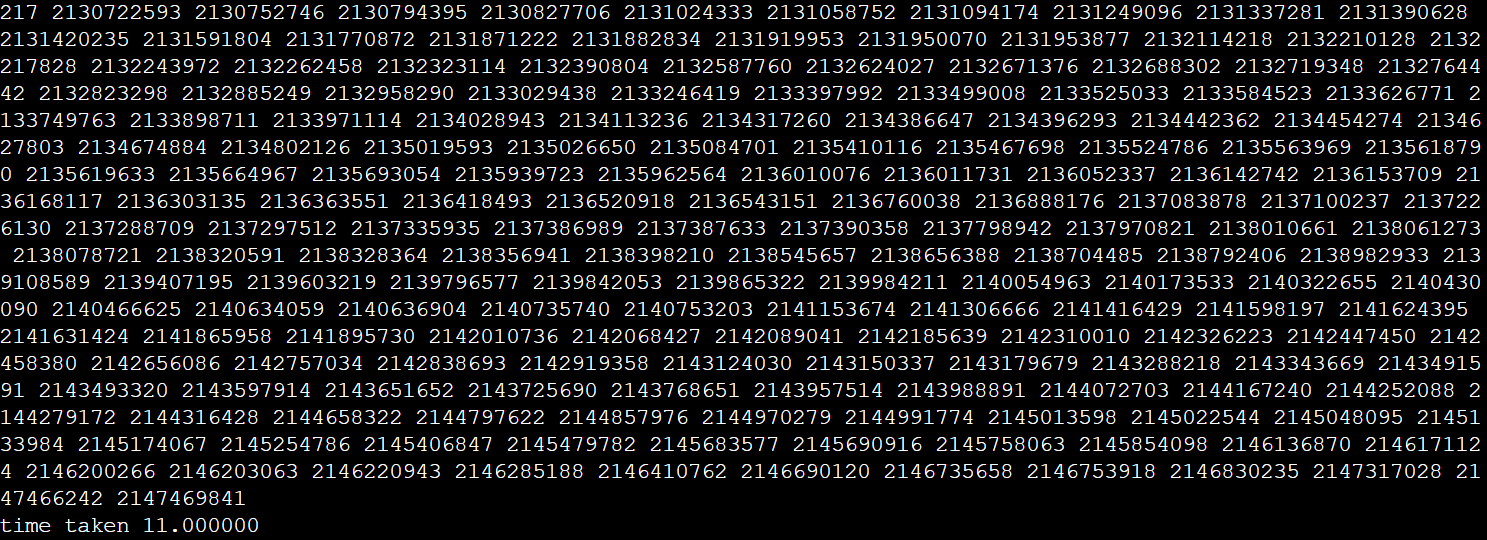
    end=time(NULL);

    printf("time taken %f", difftime(end,start));

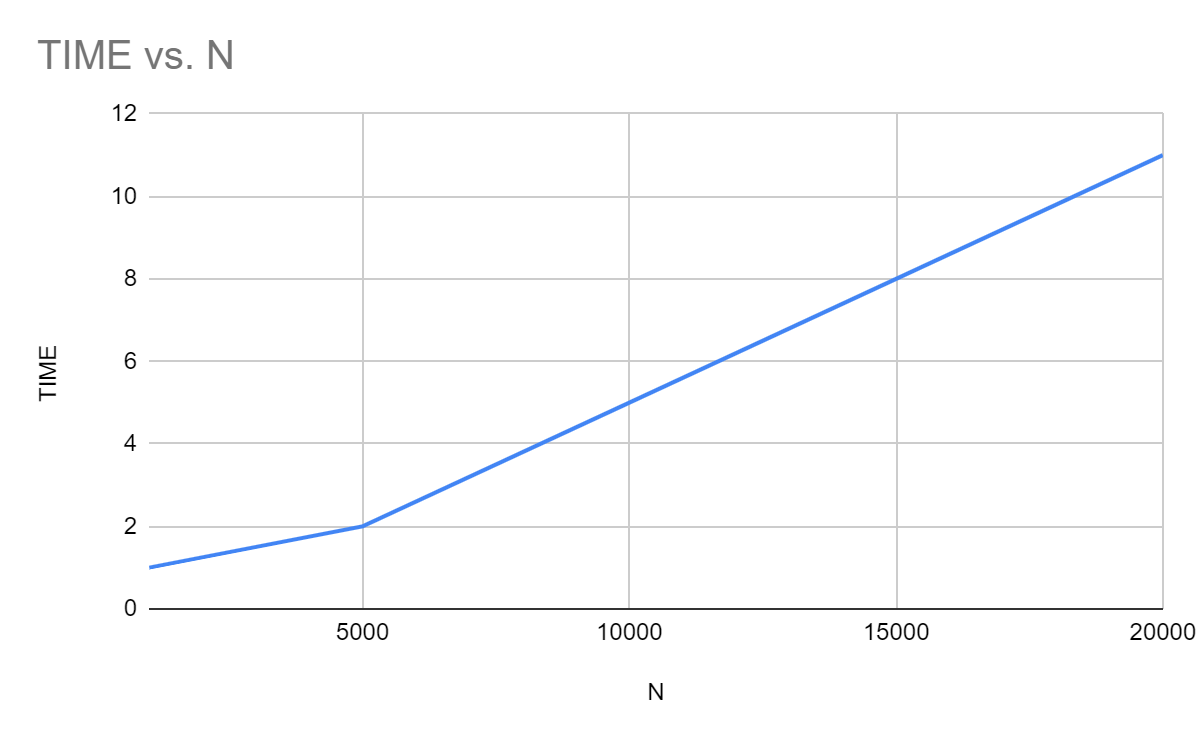
    return 0;

}

**OUTPUT:**



**GRAPH:**

****

**PROGRAM 9:** Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

#include<stdio.h>

#include <stdlib.h>

 #include<time.h>

void quicksort(int number[25],int first,int last){

   int i, j, pivot, temp;

   if(first<last){

      pivot=first;

      i=first;

      j=last;

      while(i<j){

         while(number[i]<=number[pivot]&&i<last)

         i++;

         while(number[j]>number[pivot])

         j--;

         if(i<j){

            temp=number[i];

            number[i]=number[j];

            number[j]=temp;

         }

      }

      temp=number[pivot];

      number[pivot]=number[j];

      number[j]=temp;

      quicksort(number,first,j-1);

      quicksort(number,j+1,last);

   }

}

int main(){

    time\_t start, end;

   int i, count, number[10000];

   printf("How many elements are u going to enter?: ");

   scanf("%d",&count);

   for(i=0;i<count;i++)

   {

       number[i]=rand();

   }

   start=time(NULL);

   quicksort(number,0,count-1);

   end=time(NULL);

   printf("Order of Sorted elements: ");

   for(i=0;i<count;i++)

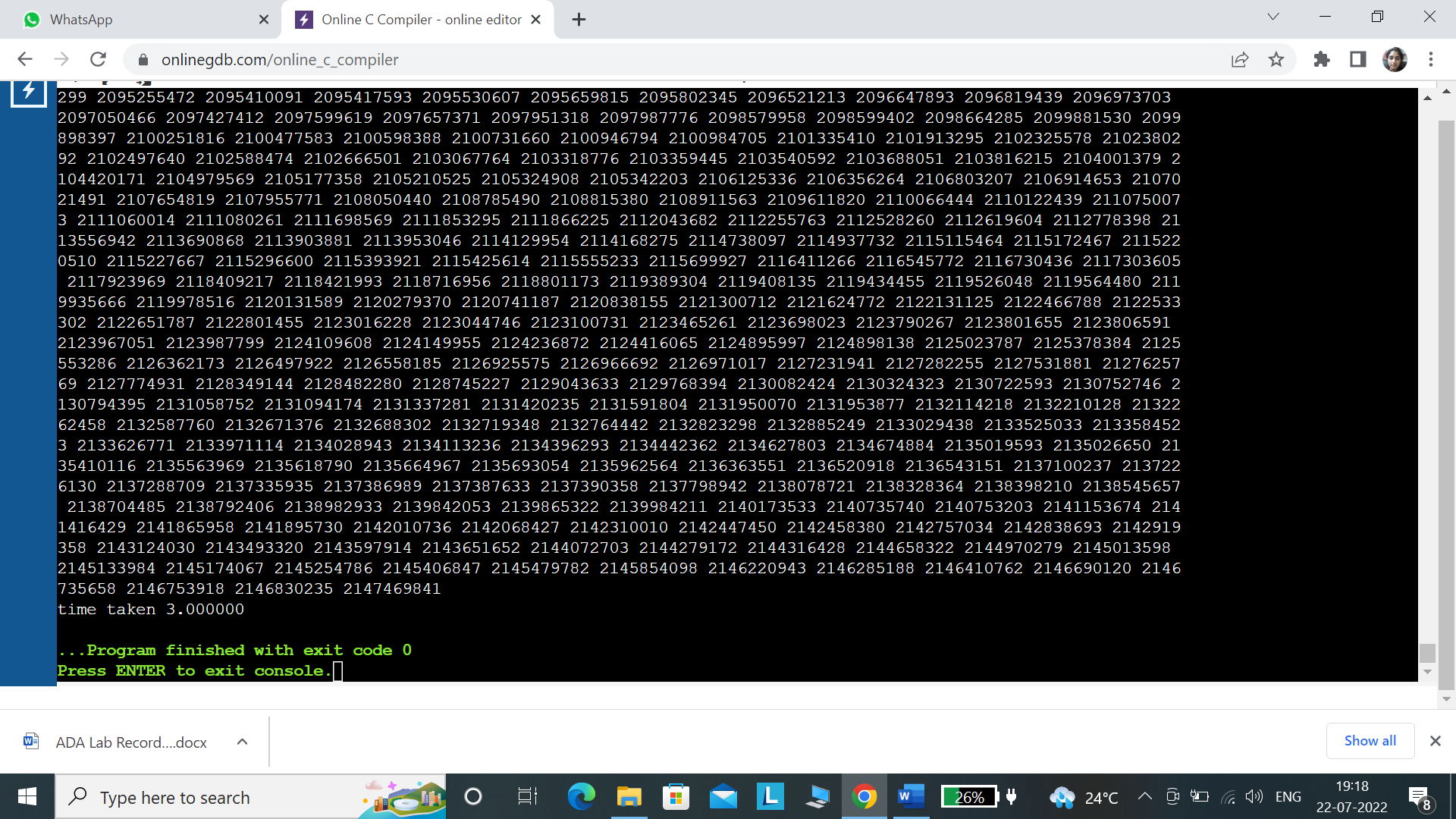
   printf(" %d",number[i]);

   printf("\ntime taken %f", difftime(end,start));

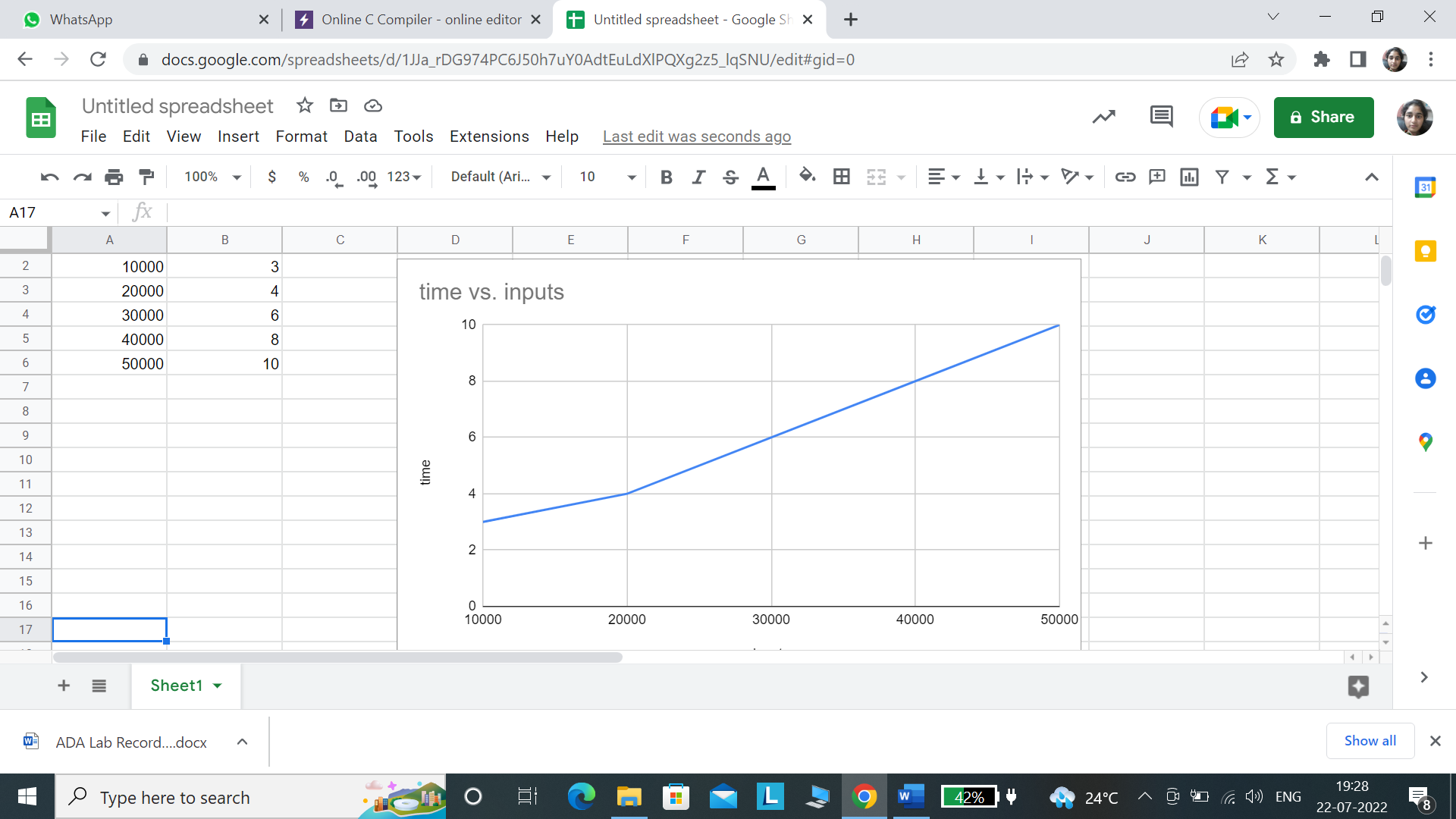
   return 0;

}

**OUTPUT:**



**GRAPH:**



**PROGRAM 10:** Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

#include <stdio.h>

#include <stdlib.h>

#include<conio.h>

#include<time.h>

void heapify(int a[], int n, int i)

{

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && a[left] > a[largest])

largest = left;

if (right < n && a[right] > a[largest])

largest = right;

if (largest != i) {

int temp = a[i];

a[i] = a[largest];

a[largest] = temp;

heapify(a, n, largest);

}

}

void heapSort(int a[], int n)

{ int i, temp;

for ( i = n / 2 - 1; i >= 0; i--)

heapify(a, n, i);

for ( i = n - 1; i >= 0; i--)

{

temp = a[0];

a[0] = a[i];

a[i] = temp;

heapify(a, i, 0);

}

}

void printArr(int arr[], int n)

{ int i;

for ( i = 0; i < n; ++i)

{

printf("%d", arr[i]);

printf(" ");

}

}

int main()

{

int i,n;

int a[100000];

time\_t start,end;

printf("enter the no.of elements");

scanf("%d",&n);

start=time(NULL);

for(i=0;i<n;i++)

{

a[i]=rand();

}

heapSort(a, n);

printf("\nAfter sorting array elements are - \n");

printArr(a, n);

end=time(NULL);

printf("\nThe time taken is %f",difftime(end,start));

return 0;

}

**OUTPUT:**

Graphical user interface, text

Description automatically generated

**GRAPH:**

Graphical user interface, chart

Description automatically generated

**PROGRAM 11:** Implement Warshall’s algorithm using dynamic programming.

#include<stdio.h>

int a[30][30];

void warshall(int n){

for(int k=1;k<=n;k++)

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

a[i][j]=a[i][j]|| (a[i][k] && a[k][j]);

}

int main(){

int n;

printf("Enter no of vertices: \n");

scanf("%d",&n);

printf("Enter adjacency matrix: \n");

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

scanf("%d",&a[i][j]);

warshall(n);

printf("Transitive Closure: \n");

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++)

printf("%d ",a[i][j]);

printf("\n"); }

}

**OUTPUT:**

A screenshot of a computer

Description automatically generated

**PROGRAM 12:** Implement 0/1 Knapsack problem using dynamic programming.

#include<stdio.h>

void knapsack();

int max(int,int);

int i,j,n,m,p[10],w[10],v[10][10];

void main()

{

printf("\nenter the no. of items:\t");

scanf("%d",&n);

printf("\nenter the weight of the each item:\n");

for(i=1;i<=n;i++)

{

scanf("%d",&w[i]);

}

printf("\nenter the profit of each item:\n");

for(i=1;i<=n;i++)

{

scanf("%d",&p[i]);

}

printf("\nenter the knapsack's capacity:\t");

scanf("%d",&m);

knapsack();

}

void knapsack()

{

int x[10];

for(i=0;i<=n;i++)

{

for(j=0;j<=m;j++)

{

if(i==0||j==0)

{

v[i][j]=0;

}

else if(j-w[i]<0)

{

v[i][j]=v[i-1][j];

}

else

{

v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);

}

}

}

printf("\nthe output is:\n");

for(i=0;i<=n;i++)

{

for(j=0;j<=m;j++)

{

printf("%d\t",v[i][j]);

}

printf("\n\n");

}

printf("\nthe optimal solution is %d",v[n][m]);

printf("\nthe selected are\n");

for(i=n;i>=1;i--)

{

if(v[i][m]!=v[i-1][m])

{

x[i]=1;

m=m-w[i];

}

else

{

x[i]=0;

}

}

for(i=1;i<=n;i++)

{

if(x[i]==1)

printf("%d\t",i);

}

}

int max(int x,int y)

{

if(x>y)

{

return x;

}

else

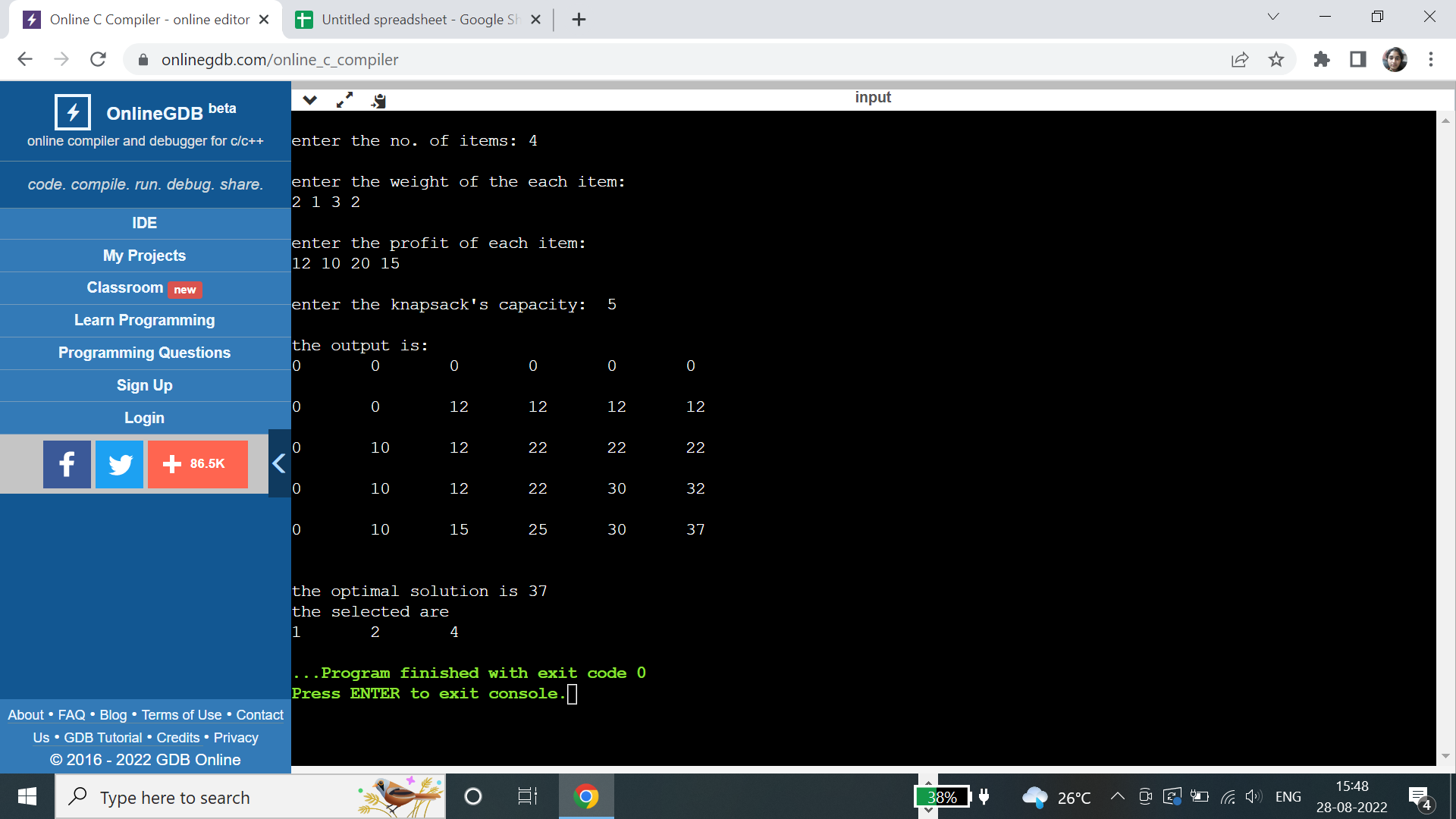
{

return y;

}

}

**OUTPUT:**



**PROGRAM 13:** Implement All Pair Shortest paths problem using Floyd’s algorithm.

#include<stdio.h>

int n;

void display(int dist[][n]);

void floyd (int graph[][n])

{

int dist[n][n], i, j, k;

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

dist[i][j] = graph[i][j];

for (k = 0; k < n; k++)

{

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

if (dist[i][k] + dist[k][j] < dist[i][j])

dist[i][j] = dist[i][k] + dist[k][j];

}

}

}

display(dist);

}

void display(int dist[][n])

{

printf ("DISTANCE MATRIX \n");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (dist[i][j] == 99)

printf("99 ");

else

printf ("%d ", dist[i][j]);

}

printf("\n");

}

}

int main()

{

printf("ENTER ORDER OF MATRIX \n");

scanf("%d",&n);

int graph[n][n];

printf("ENTER ELEMENTS OF MATRIX and 99 FOR INFINITY\n");

for(int i = 0;i < n;i++)

{

for(int j = 0;j < n; j++)

{

scanf("%d",&graph[i][j]);

}

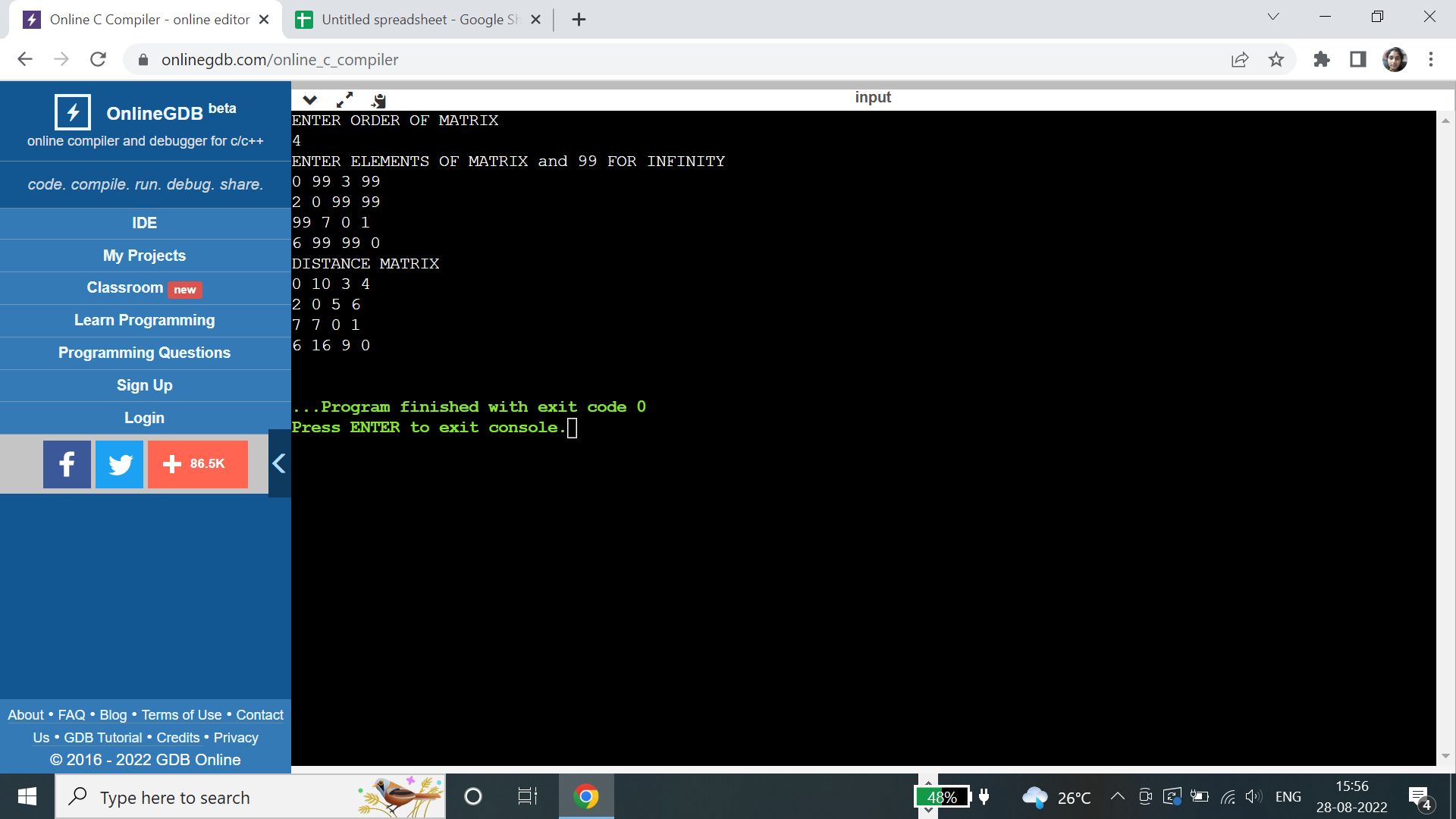
}

floyd(graph);

return 0;

}

**OUTPUT:**



**PROGRAM 14:** Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.

#include<stdio.h>

void prims();

int c[10][10],n;

void main()

{

int i,j;

printf("\nenter the no. of vertices:\t");

scanf("%d",&n);

printf("\nenter the cost matrix:\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&c[i][j]);

}

}

prims();

}

void prims()

{

int i,j,u,v,min;

int ne=0,mincost=0;

int elec[10];

for(i=1;i<=n;i++)

{

elec[i]=0;

}

elec[1]=1;

while(ne!=n-1)

{

min=9999;

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

if(elec[i]==1)

{

if(c[i][j]<min)

{

min=c[i][j];

u=i;

v=j;

}

}

}

}

if(elec[v]!=1)

{

printf("\n%d----->%d=%d\n",u,v,min);

elec[v]=1;

ne=ne+1;

mincost=mincost+min;

}

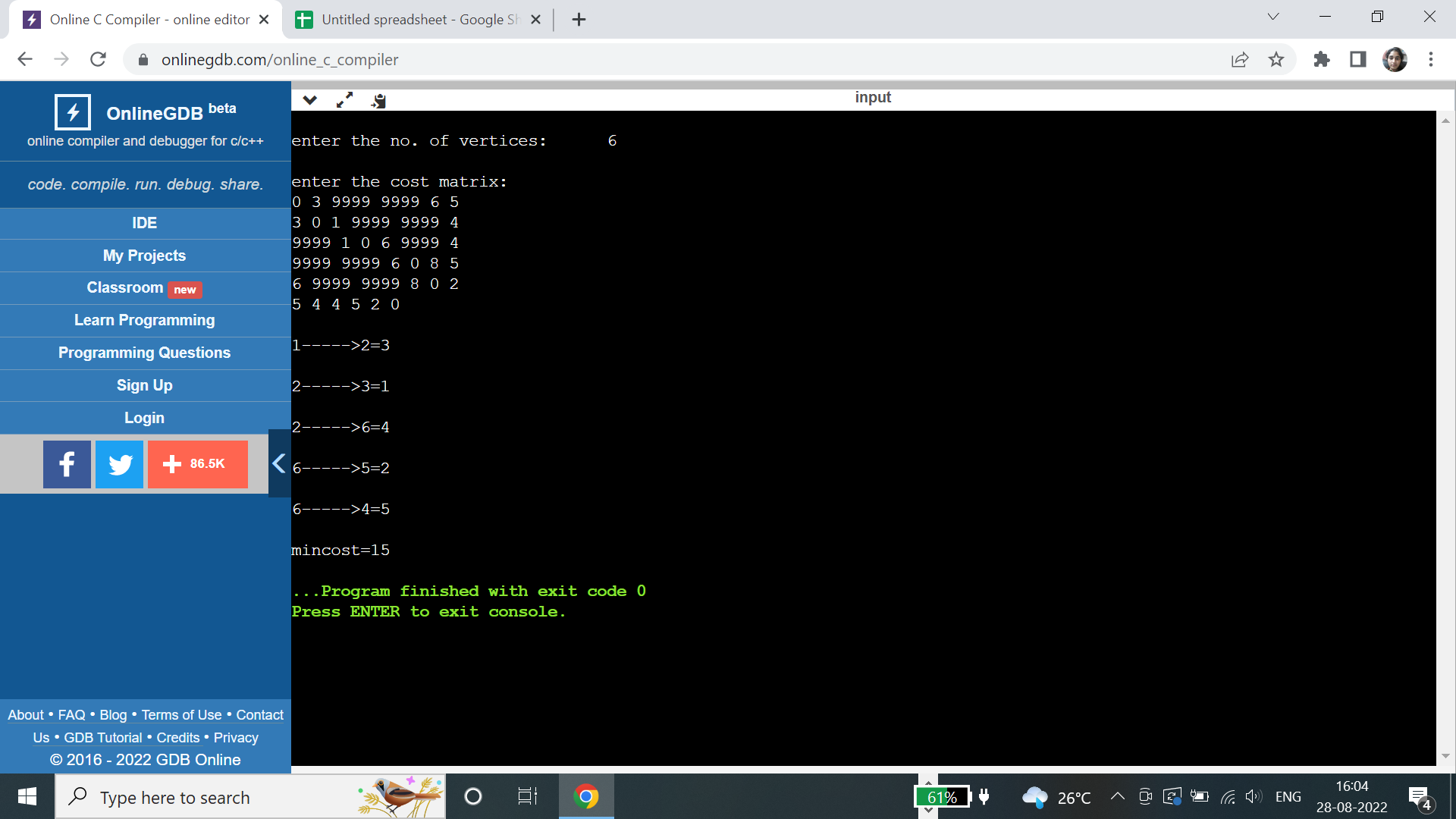
c[u][v]=c[v][u]=9999;

}

printf("\nmincost=%d",mincost);

}

**OUTPUT:**



**PROGRAM 15:** Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.

#include<stdio.h>

#include<conio.h>

void kruskals();

int c[10][10],n;

void main()

{

int i,j;

printf("\nenter the no. of vertices:\t");

scanf("%d",&n);

printf("\nenter the cost matrix:\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&c[i][j]);

}

}

kruskals();

getch();

}

void kruskals()

{

int i,j,u,v,a,b,min;

int ne=0,mincost=0;

int parent[10];

for(i=1;i<=n;i++)

{

parent[i]=0;

}

while(ne!=n-1)

{

min=9999;

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

if(c[i][j]<min)

{

min=c[i][j];

u=a=i;

v=b=j;

}

}

}

while(parent[u]!=0)

{

u=parent[u];

}

while(parent[v]!=0)

{

v=parent[v];

}

if(u!=v)

{

printf("\n%d----->%d=%d\n",a,b,min);

parent[v]=u;

ne=ne+1;

mincost=mincost+min;

}

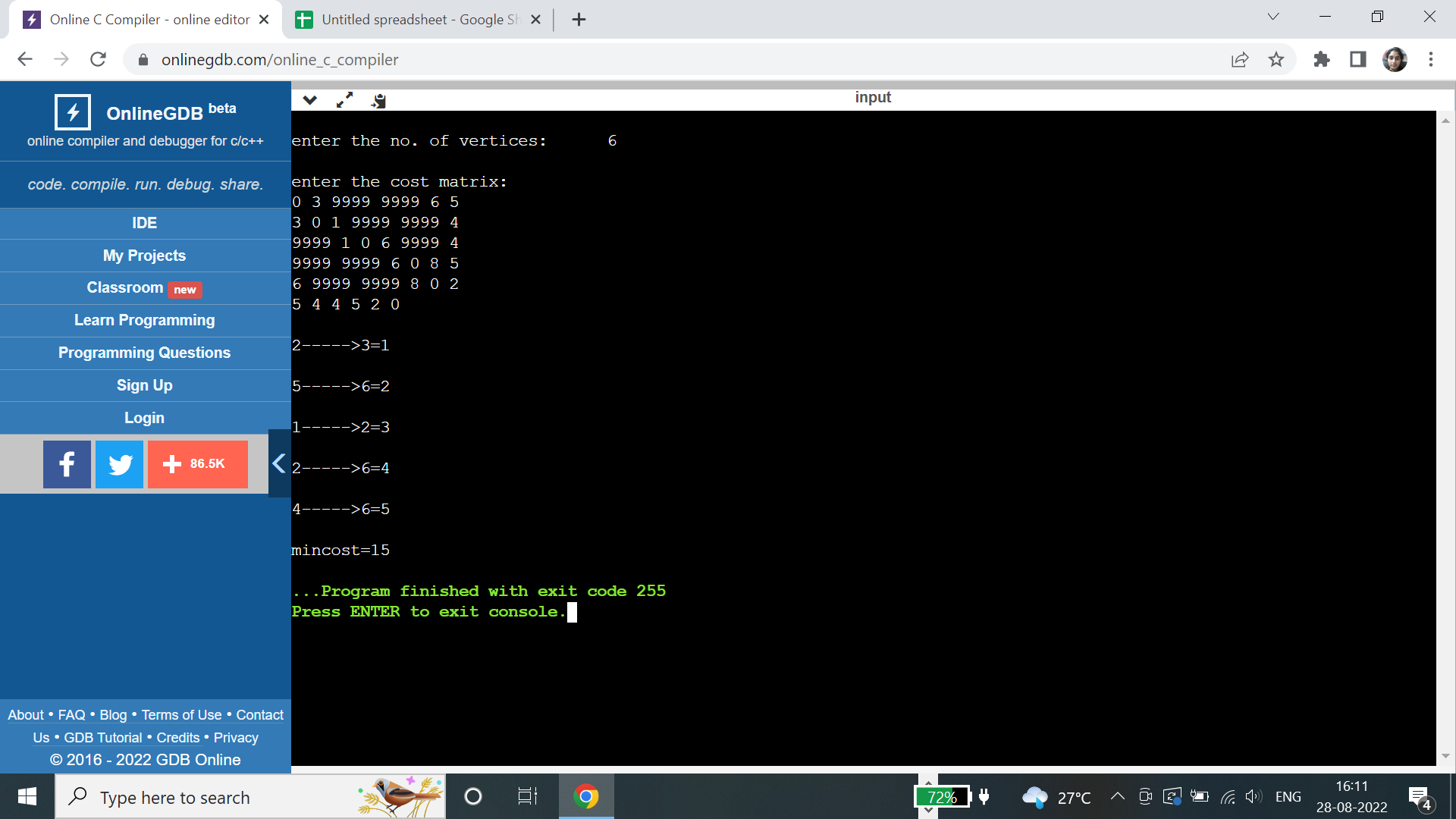
c[a][b]=c[b][a]=9999;

}

printf("\nmincost=%d",mincost);

}

**OUTPUT:**



**PROGRAM 16:** From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm.

#include<stdio.h>

void dijkstras();

int c[10][10],n,src;

void main()

{

int i,j;

printf("\nenter the no of vertices: ");

scanf("%d",&n);

printf("\nenter the cost matrix:\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&c[i][j]);

}

}

printf("\nenter the source node: ");

scanf("%d",&src);

dijkstras();

}

void dijkstras()

{

int vis[10],dist[10],u,j,count,min;

for(j=1;j<=n;j++)

{

dist[j]=c[src][j];

}

for(j=1;j<=n;j++)

{

vis[j]=0;

}

dist[src]=0;

vis[src]=1;

count=1;

while(count!=n)

{

min=9999;

for(j=1;j<=n;j++)

{

if(dist[j]<min&&vis[j]!=1)

{

min=dist[j];

u=j;

}

}

vis[u]=1;

count++;

for(j=1;j<=n;j++)

{

if(min+c[u][j]<dist[j]&&vis[j]!=1)

{

dist[j]=min+c[u][j];

}

}

}

printf("\nthe shortest distance is:\n");

for(j=1;j<=n;j++)

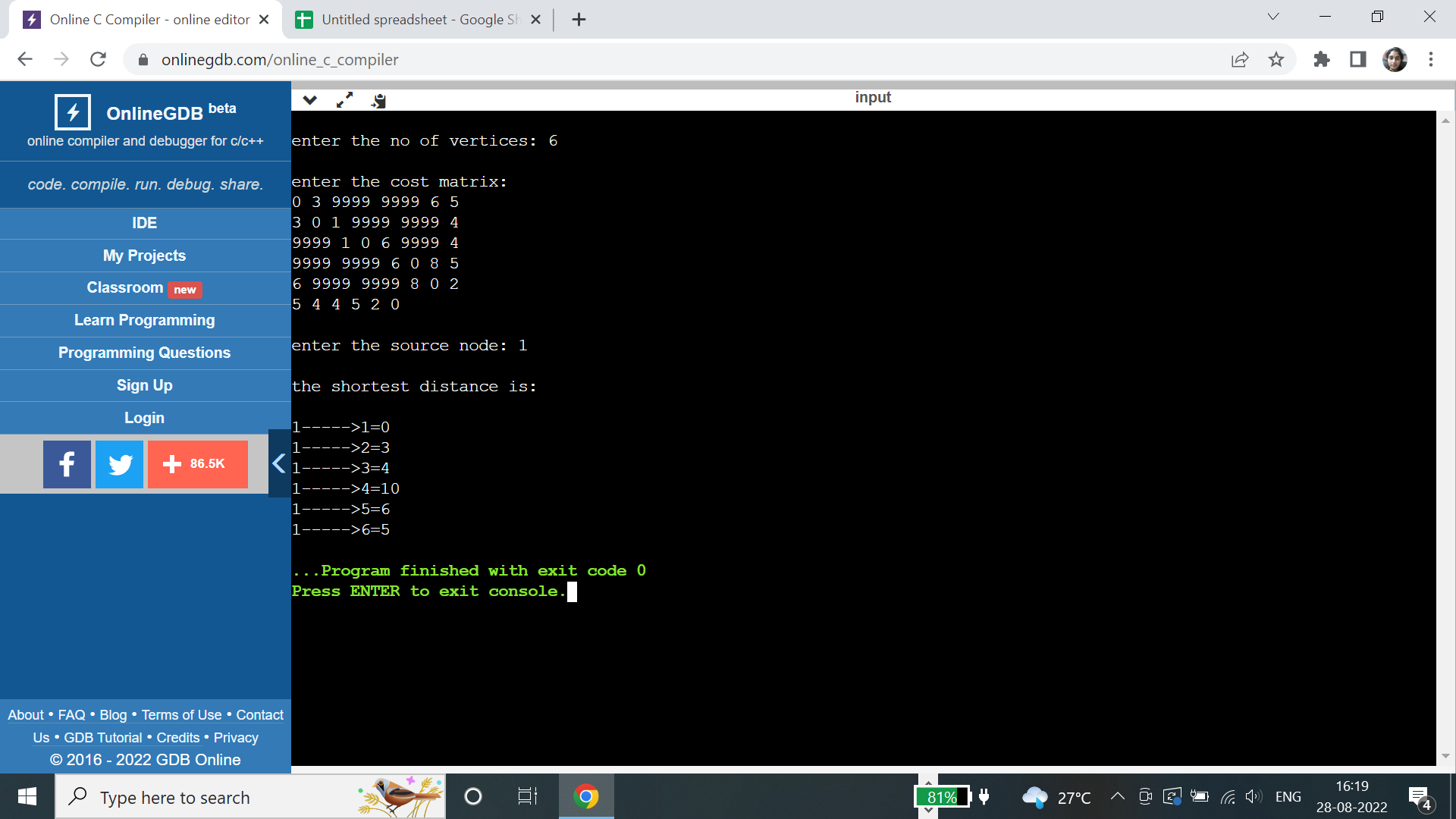
{

printf("\n%d----->%d=%d",src,j,dist[j]);

}

}

**OUTPUT:**



**PROGRAM 17:** Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t have a solution.

#include<stdio.h>

#include<conio.h>

int s[10] , x[10],d ;

void sumofsub ( int , int , int ) ;

void main ()

{

int n , sum = 0 ;

int i ;

printf ( " \n Enter the size of the set : " ) ;

scanf ( "%d" , &n ) ;

printf ( " \n Enter the set in increasing order:\n" ) ;

for ( i = 1 ; i <= n ; i++ )

scanf ("%d", &s[i] ) ;

printf ( " \n Enter the value of d : \n " ) ;

scanf ( "%d" , &d ) ;

for ( i = 1 ; i <= n ; i++ )

sum = sum + s[i] ;

if ( sum < d || s[1] > d )

printf ( " \n No subset possible : " ) ;

else

sumofsub ( 0 , 1 , sum ) ;

getch () ;

}

void sumofsub ( int m , int k , int r )

{

int i=1 ;

x[k] = 1 ;

if ( ( m + s[k] ) == d )

{

printf("Subset:");

for ( i = 1 ; i <= k ; i++ )

if ( x[i] == 1 )

printf ( "\t%d" , s[i] ) ;

printf ( "\n" ) ;

}

else

if ( m + s[k] + s[k+1] <= d )

sumofsub ( m + s[k] , k + 1 , r - s[k] ) ;

if ( ( m + r - s[k] >= d ) && ( m + s[k+1] <=d ) )

{

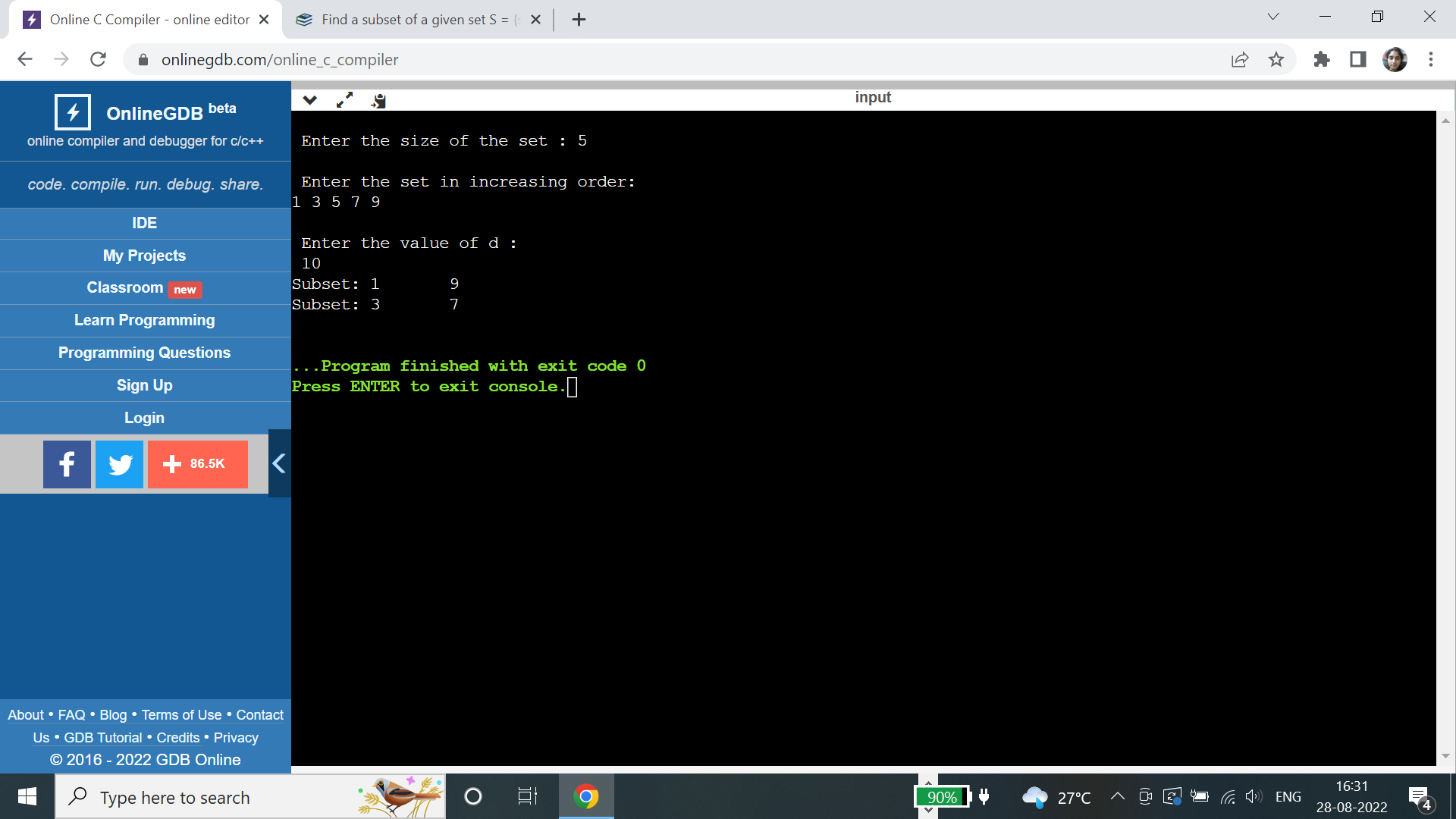
x[k] = 0;

sumofsub ( m , k + 1 , r - s[k] ) ;

}

}

**OUTPUT:**



**PROGRAM 18:** Implement “N-Queens Problem” using Backtracking.

#include<stdio.h>

#include<math.h>

int board[20],count;

int main()

{

int n,i,j;

void queen(int row,int n);

printf("\n\nEnter number of Queens:");

scanf("%d",&n);

queen(1,n);

return 0;

}

void print(int n)

{

int i,j;

printf("\n\nSolution %d:\n\n",++count);

for(i=1;i<=n;++i)

printf("\t%d",i);

for(i=1;i<=n;++i)

{

printf("\n\n%d",i);

for(j=1;j<=n;++j)

{

if(board[i]==j)

printf("\tQ");

else

printf("\t-");

}

}

}

int place(int row,int column)

{

int i;

for(i=1;i<=row-1;++i)

{

if(board[i]==column)

return 0;

else

if(abs(board[i]-column)==abs(i-row))

return 0;

}

return 1;

}

void queen(int row,int n)

{

int column;

for(column=1;column<=n;++column)

{

if(place(row,column))

{

board[row]=column;

if(row==n)

print(n);

else

queen(row+1,n);

}

}

}

**OUTPUT:**

