We've come a long way in this course and learned:

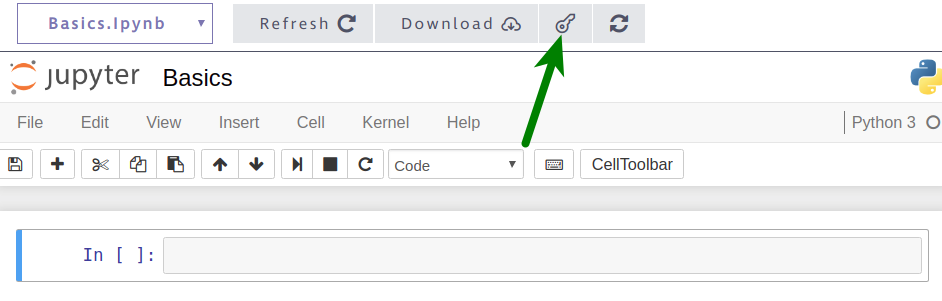
* The basics of programming in Python (arithmetical operations, variables, common data types, etc.)
* List and for loops
* Conditional statements
* Dictionaries and frequency tables
* Functions
* Jupyter Notebook

To make learning smoother and more efficient, we learned about each of these topics in isolation. In this **guided project**, we'll learn to combine these skills to perform practical data analysis. Guided projects are more involved compared to regular missions, so you should expect to spend more time working on them. After completing your work, you should be able to add a project like [this](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb) to your portfolio.

For this project, we'll pretend we're working as data analysts for a company that builds Android and iOS mobile apps. We make our apps available on Google Play and the App Store.

We only build apps that are free to download and install, and our main source of revenue consists of in-app ads. This means our revenue for any given app is mostly influenced by the number of users who use our app — the more users that see and engage with the ads, the better. Our goal for this project is to analyze data to help our developers understand what type of apps are likely to attract more users.

This is intended to be a portfolio project; you may want to share it with potential employers one day, so we'll start our work by helping readers understand what this project is about and what we're trying to achieve. You'll be able to check the solutions for this project in this [solution notebook](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb). The key icon at the top of our interface will also take you to the solution notebook.



To follow the instructions, you can use the Jupyter Notebook interface you see on the right of the screen.

Instructions

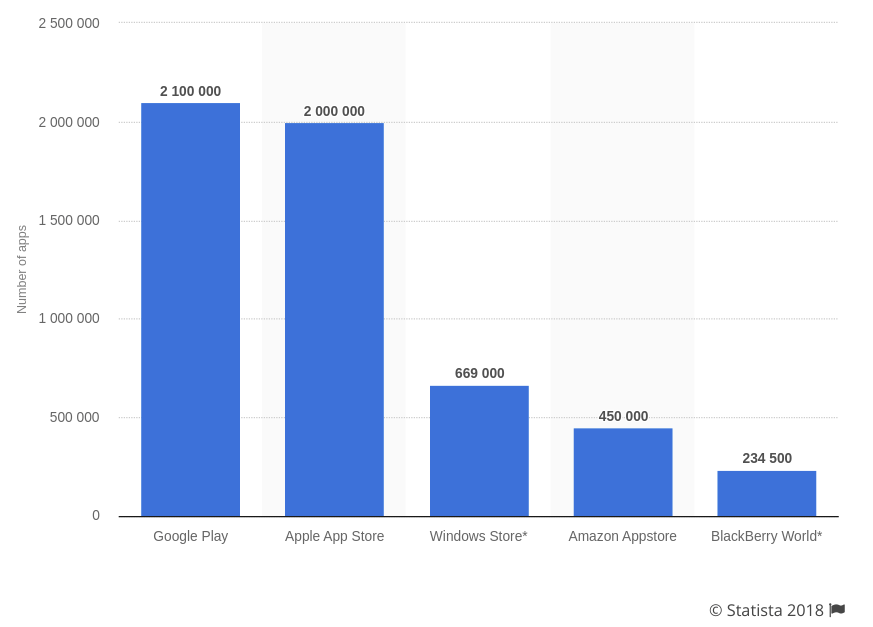
To help readers gain context into your project, use the first Markdown cell of the notebook to:

1. Add a title.
2. Write a short introduction where you explain (in no more than two paragraphs):
   * What the project is about.
   * What your goal is in this project.

The title and the introduction are tentative at this point, so don't spend too much time here — you can come back later to refine them.

In the previous step, we outlined that our aim is to help our developers understand what type of apps are likely to attract more users on Google Play and the App Store. To do this, we'll need to collect and analyze data about mobile apps available on Google Play and the App Store.

As of September 2018, there were approximately 2 million iOS apps available on the App Store, and 2.1 million Android apps on Google Play.

Source: [Statista](https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/)

Collecting data for over 4 million apps requires a significant amount of time and money, so we'll try to analyze a sample of the data instead. To avoid spending resources on collecting new data ourselves, we should first try to see if we can find any relevant existing data at no cost. Luckily, these are two data sets that seem suitable for our goals:

* [A data set](https://www.kaggle.com/lava18/google-play-store-apps) containing data about approximately 10,000 Android apps from Google Play; the data was collected in August 2018. You can download the data set directly from [this link](https://dq-content.s3.amazonaws.com/350/googleplaystore.csv).
* [A data set](https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps) containing data about approximately 7,000 iOS apps from the App Store; the data was collected in July 2017. You can download the data set directly from [this link](https://dq-content.s3.amazonaws.com/350/AppleStore.csv).

We'll start by opening and exploring these two data sets. To make them easier for you to explore, we created a function named explore\_data() that you can repeatedly use to print rows in a readable way.

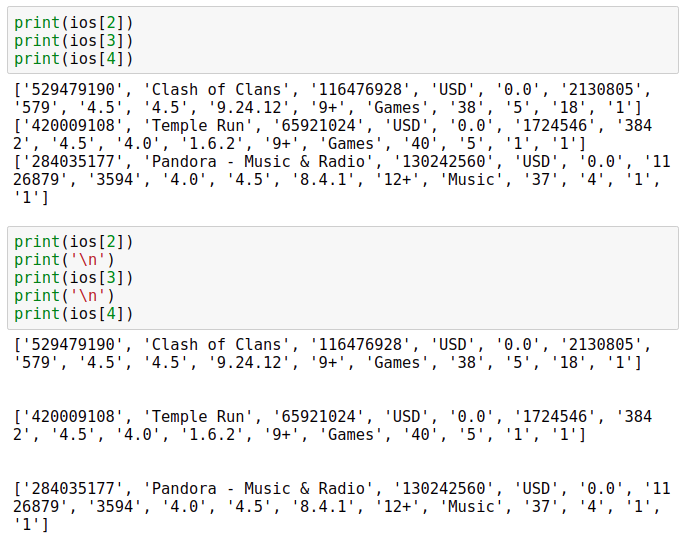
![A screenshot of a cell phone

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAeAB4AAD/4RDyRXhpZgAATU0AKgAAAAgABAE7AAIAAAANAAAISodpAAQAAAABAAAIWJydAAEAAAAaAAAQ0OocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE1hdHRoZXdTcGlyZQAAAAWQAwACAAAAFAAAEKaQBAACAAAAFAAAELqSkQACAAAAAzk0AACSkgACAAAAAzk0AADqHAAHAAAIDAAACJoAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIwOjA1OjE0IDEyOjM5OjQxADIwMjA6MDU6MTQgMTI6Mzk6NDEAAABNAGEAdAB0AGgAZQB3AFMAcABpAHIAZQAAAP/hCx9odHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDIwLTA1LTE0VDEyOjM5OjQxLjkzNjwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5NYXR0aGV3U3BpcmU8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgA/wNvAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaTeo/iH50tVH/1jfU01qItb1/vD86N6/3h+dVKKdguW96/3h+dG9f7w/OqlFFguW96/wB4fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev8AeH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/AHh+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/wB4fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev8AeH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/AHh+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/wB4fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev8AeH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/AHh+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/wB4fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev8AeH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/eH50b1/vD86qUUWC5b3r/AHh+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/3h+dG9f7w/OqlFFguW96/wB4fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlvev94fnRvX+8PzqpRRYLlsMD0IP0paht/4vwqapGFVH/wBY31NW6qP/AKxvqaaExKKKKoQUUVS0/U4dSe7WBZFNpcNbvvAGWABJGD05our2Cztcu0UVn6hq8dhdW9qtvPdXVxuMcEAXdtUcsSxAAGQOT3pNpbjSbdkaFFRwSNNAskkMkDMOY5CpZfrtJH5GpKYgooooAKKKKACise08QrceIn0afTry0uVgNwrTGIpIgbbkFHY9T0IFbFMQUUUUhhRRRQAUUVn6jrFvpl3p9vOkjPqFx9niKAEK21my2T0wp6ZpiNCiiqupalaaRp8t9qEvk20IBd9pbGTjoAT1NAy1RSdaWkAUVn61q8Gh6W97cq8mGVI4Yxl5pGOFRR3JJAq3bvLJaxPcRCGZkBkjD7gjY5Ge+D3oAlorPfWLdPEUWilJPtMlq10GwNmxWCkZznOWHatCjpcAoqpfanaaa1sL2Xyzdzrbw/KTvkYEheBx0PJ4q3TAKKKKQBRRRQAUVU0/UrTVbd5rCXzY45XhZtpXDoxVhyB0IIq3QAUUUUAFFFVbrUrSyurS2uZdkt7IYoF2k72CliMgccAnmmBaooopAFFFVb7UrTTRAb2XyxcTrbxfKTukb7o4HH1PFAFqiiigAooooAKKKKACikqjourwa7pMWoWiSJDKWCrKAGG1ip6E9xQBfooooAKKKKACiiigAooooAKKKKACiiigAorP1fWLfRYbaS6SR1ubqK1TywDh5G2gnJHGetaFABRRRQAUUVkeIdfHh3TnvptOvLu2iRnmktjF+6UdyHdSfwz0oA16KbG4kjV16MARmnUxbhRRRSGFFFFABRRVW/1G20y3E147IjOsa7I2clicAAKCTmhtJXYJNuyLVFVLPUYb5mEKXKlRk+fayxflvUZ/CrdCaaugaadmFFZmsa0ujQieeyuZbfKhpovLwhZgoBBYHqR0BrTpJpjaaVwooopiCisvX9aGgaet/NaSz2qSAXLxEZt4z1kI6kA4zjnGT2rSR1kjV42DIwBVlOQR60wHUUUUgCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAJrf+L8KmqG3/i/Cpql7jQVUf/WN9TVuqj/6xvqaEDEoooqhGBeSSX/i1NKknmitY7P7SywSNG0jF9oyykNgAdAR15rDtL19H8N+JrjTnZni1J0jdn8wrnYu4ljyRnPJ7c1119pNnqMkctyj+ZECEkimeJgD1G5CDg46dKLbR9Ps7ae3t7SJILhi0sWMo2QB908YwAMdKw9nK7fk/wAX+huqkUkvTT0/zMSyg1K31Wxa0tNVW35S8a/vElV128MB5jENux90AcmobmwuT4+tUGr3gZrGVg4SHKjevyj93jH1BPvW9Z6LZ2EiNbNdKIxtVGvJnQDGMbGYr+nFWGsbd9RS+aPNzHGYlfceFJyRjp1FNU9Ffv8A1tYPaK79P67ksSNHCiPI0rKMGRwMt7nAA/ICqmtXsmm6Df3sEfmSW1vJKif3iqkgfpV6kIDKQwBBGCD3rZ6oxT1uzz3TRrslvpN/p9nrst1LJFJd3F1fwtbzxOcyERCZgowcrsQEYA9am8a6vf8AhXVlvtKumuDfQMk1jMzyLbhB/wAfSqA21Vz8wAAbjvXS2nhbS7CZJLFbu3WN96QRX86wqc5wIg+wDJ+7tx7VZi0TT4tVutSW33Xd3GsU0kjs+UHRQGJCrz0GAe9aSkm/6+70M4ppa/15nJ63Bdaf4e8N2um6/eu93qcCSagJ/MeZXVixG7I2nqByo444rS05X0rx1JpNtc3k9nLp/wBpZLm4ecxSCTaCHclgGGeM4+XtUOueEUOl6Jpmi28y2drqkc8iJdurRRANuKsW3DGRgKeO1dBpujWOkmZrOJ/NnIMs00zzSyY4ALuSxA7DOBk4ourX83+SCz0XkvzZit/yVqP/ALArf+jhSzb9b8a3ul3VzdQWljaQypFbXLwGV5C4LFkIYgBcYzjkkjpVv/hDtIOofbs6j9q27PN/tW63bc52/wCs+7nnHSreo6Bp+qXMdzdRSrcxKUSe3uJIJAp6rvjZSVzzgnFK6sv67lO9352/T/I42+1PUbb4e+Ko4765eTSrh7e1vC5EpUBCMt1JBYru6nHPOa6az8PXFrqNlfR61fyMquLyO4laRLrcOCEzsjIYAjYo4yO9XH8P6XJoL6MbNRp8ilXhViu4E5JLA5JJ5JzknkmtEDAAHQUc3YLfr/XyPNorrWtd07Ub+0s9dfUftE6WcttfQx28BjcqimIzAMPl+behJye2K9EtHnksYHu4xFcNGpljBzsbHIyOuDWbN4X0ua7luVS6tpJ23y/Y76a3WRv7zLG6gt7kZqW81O7s7jybfQtQvkCjE0EluFPt+8lVs/hRe6sK2tyLxZqk+i+EdT1G0AM9vbs8e4ZAbsSPbrXNaxpUGn+IPCUn228ubqXUCXa4vJJBJ+5fcwjLbF5I+6oAzgYHFdVbXcuqia11DQry0geMhjdtA6SA8FcJIx5B7jFQW3hLRbSW2litZGktXDQSTXMsrR4UqFBZiQoBPy/d5zihe6wfvK3qY+vXUOpX+o2tnDr93PZxhJJNNvRbR2zsuQOZYw7Yw3IfGR9Kx9fLa98F7fV9RmuGu2somYx3Dxo7FlyWRSFb15HHauxufCmjXl/PeT2rGS52/aFWeRY58DA8yMMEfjj5gatxaLp0Ohro6WqHT1i8kW75ZdnpzkmhSSX3f8Edry/ryGWen2mi280yTXZTZuka5vJrjaFBOR5jNj8OtcPcarJG2i6tpUXiBYrvUIIzd3l4rQXUUjbSfJMh27gdwxGuOOnSu5sNEs9Nhlhg+0yRSqEZLq7luF2jIwBIzYHPbrVCLwRoEIt1FnI62siS2yy3UsiwMpyPLDMQgyBwuAeh4pppSu/L/gk2fLb+vIr67ibx34ZgnGYV+0zqCeDKqALx3IDMR/8AWqfxff3NnZ6fb2czW7ahqENo86AFo0YksRngEgYB7Z9al8TaPcanZ29xpkkcWqWEwuLOSQHaWAIKNjnaykqfrntV25sYda0dbfV7QASqrSQ+YcxuMHh1wQQRwwweMis1t8/6/wAi3+n9f5nK2mn2um/Fy3htZrmQnRZWcXN7LcMv76PH+sZiBwelZniDWSdJude0KLxA5huAU1A3iraviQKy+Q0oynVf9Vk4zkn5q7ax8N6Vp18t7a27faxG0ZuJJnkkdWKk7mZiWPyryckAADjiqT+BPDssUkMtgz28jtJ9na5lMKOTuLJHu2ock8qARk+pprS3l/ncT1v/AF0Mnx7pNvc3/h6eSS8V5tXgiYR3syKF2vyqqwCt/tAA+9dBd/YfDPhy7mlu7yK2iUu80k8lzKpPHymQuc5xgdM9qu6jpdnq1kbXUIfNi3BhhirKwOQysCCpB6EEEVAdA099Gn0q4Sa6s7gESpd3Mk5YH/adiw9sHg8il9lpf1sP7Sb/AK3/AMzk7G6urHxro0ENlrljbagkyyx6pfC5WTam5SoMshVh/wAB4Pen66dQ03XNS1LXIdXuNJRY3tbnSr0otmgGG8yAOpfDZYnbINvGMAiuitfCej2d/b30dvLJeW2RFcXFzLNIgIwV3OxO3BPy9ASTjNOvfDGl6hcSS3UdwwlOZYVvJkhl9d8SsEbPfIOe+ab30EvMw/EurT3Ws6Lp9jHqFzY3tvLdONOnSCWcLs2gOzoVX58nawJ4HTNWfD9jqU8WqWOrW+qWumtIjWRub8G5VSMsvmxSM+Aw4JbOGx0Fbeo6Lp+qwRRXkGfIbdC8TtFJCfVHQhl444IyOOlP0/TINMhaO2kunVm3E3N3LcHPsZGYgew4o01/r+v67hrocj8PdDtF064uhLf+ZFql2qqdQnKHEzgZTftY+pIOTycmuyub60s3hS7uoYGuHEcKyyBTI/8AdXPU+wqlF4c0231R7+2S4gmklM0iQ3cqRO5GCzRBghJ7krz1NZ3ifw1L4n1HTI7hLeKxsLlLwTh2MzOuf3YXaAqnjLbj0xt70LZL0/4IbXfqdNRRSUhnBWNxeaHqkLeLo9XW5nvTDHqUN4ZbKTef3amENiPOQnMY553ZOam8YaLa3Hi7w1JJLfBrm+dXEd/OgAED/dCuAh46rgnnPU10EPhXSYLqKdYZ28l98MMt3LJDER0KRMxRcdsKMdsVb1PSbLWLZIdQiMixyCSNkkaN43HRldSGU+4I701ol5C6vzKOpy2PhrwzM0txfLApChlmeed2ZgAis5ZssSFHPGeo6jn9Hubqx+IFrYpbazZWl5YSu8GqXy3IZ42TDofNkK8OQRkA8ccV1E3h7TrnRX0m7ilurNzkrc3Mkr5zkHzGYsCDyDnjtUdl4X0mw1JNQgt5GvY0aMXM9xJNIUbGVLOxJHAwD05xjJoWjv8A1sD2t/W5zek6Vd+IZPEZutc1OMw6lPBZiG6eNbbhcNhSC/J+6xKjHAGTlfG2jJJD4fe9urua5OpWsEskV1LCj9csI1farE85HI9a7Cy020043Js4vLN1O1xN8xO6RsZPJ46DgcUmo6baatYtaahCJoWIOMlSpByGDDBUg8gggiktFHyt+APW/nf8Q0/TodMt2htnuXVm3E3N1JO2f96RmIHHTOKfbX9petMtndQ3DW8himEUgYxuOqtjofY1FZ6eml2ckViZ5iSXUXd5LMS2Om9yzAcdunXFZPhbwy2iXur6jceSl3q9wJ5YLckxQ4GAAxALEkklsDJPQYp7sOh0dcn4z1C7i1DRdMtY714r+aTz1sZUimkVE3eWrs67cnkkMDhTjrXWVT1LSrLV7ZYNQg81UcSRsGKPG46MrqQyt7gg0hnMad/bNnBroltdSs9NFp5lmdQu0mmjl2tvAdZHbHCkbm4OccVl2JvbHw34R1ltW1C5vr+e1juWnuXaORJUwV8rOzjj5sbsjJJJOe2tdEs7O2uLdGu5Yrhdsgub2afIwRgGRyV69sUf2Dpp0+wsfs3+jac0b2qeY37sxjCHOcnHvn3prR39PwvcTWlvX/gGKgPiDxRrdrf3d3Da6WYo4oba6ktuWj3s7NGys2c4AJwNvTJNHw2KD4e6eUbdHmXDbt2R5r857/Wta+8N6VqV8bu8tmeVkEcm2Z0SZBnCyIpCyDk8MCOTUiWA0nRTaaFZwtsyY4J52RDubLZba5HUnofTilsh7stWd9aajarc6fcw3UDEhZYJA6nBwcEcdanrB8G+GI/Cfh8afHIsjNK88hRdqBmOdqg5IUDAGfSt6mxI5DV47m/+JFlpv9o3ltYtpcs00NtMY/NIlQD5hyvXqpB7ZwTSasmoeGPDbwQarcTC61KKGC4mO+SzillVSN7li5XLYZvUeldK2mWjaumqNFm8jga3WXceIywYjGcdQOcZp97ZW2pWMtnfwJcW8y7ZIpBkMKS0SX9b3/LQfX+u1jmZbd/D3izRLfT7y/mg1Eyw3EF1dyXIwkZcSAyMSpBABwQDu6cCsnR/Ft9p2n6da3bG5iha4t7+6uHJlRxJIkJyeMExsDn1Wuw07w9pul3JuLaOZ59nlia5uZbh0T+6rSMxUeoGAcD0qvL4N0Caz1O1l05Gh1aXzr1d7/vXznOc5HPPGKHdoEc+lt4gufBXhmSNry+Xy0k1KCG9MFzOrLkbZSQflJyRuXOMZroPCuo2l/ps0dn9vRrWdoZrfUWZpoHGDtLMSWGCCDubg9ewuXmh2N9DBHKk0Yt12xPbXEkDovHy70YNg4GRnBwPQUxLaHw/pjLpOmXF1mTc0cMitLIx6uzyuNx9SWJqm1dvuTZ2VzTri9fuodT1DUbWzg8Q3k9lGEkk0y9FtHauy5A5ljDtjDch8ZA9q2V13USwB8KauoJ6mWz4/wDI9OufCejXmoT3k9q5kuQouEW4kWO4wMDzIwwR+OPmB4qGmyk7HKzapqmreEfBM41K4tLnUbmFbme3IVnBictxjbzj0wDzjgVdjvZ/D2u+I7S2murm1s9Kjv4orq4kuGWT94CAzsWwdi8Zx6YzXRx+HtLis9OtY7XbBpjq9onmN+6YAqD154J65qwul2a6pNqIhH2qeFYJJCxO5FJIGM46se3enLW9ut/y0/ES0tfpb89fwPP9YsU/4R7wvql5qF3c395qVhI7SXknlyMzhjth3eWMdgF4A/GvS6wYvBegw+XtsmYQyJJAr3EriAqwYCMFiEXIB2rgHAyOBU0+tX8NxJHH4Z1SdUYgSxy2oVx6jdMDg+4BqrrVLvf8v8gSel+3+ZD4y1K60zw60lhL5E81xDbi42hvJEkioXweOAT14zirenaLDpLF7e61CbKbXW6vZJw57N+8Y7T1+7gc8jgYWFhrlhPb6tos1tC/yPBfeS4lH0R3GPril03QrHSXL2n2lm27Abm8luCi+i+YzbRwMgYzgZ6Cl3A40G5vvh3c+LJNWv4NTEEt3GY7pxDCUJKxeSD5bKNu07lJPOTnpseNJnufhVqk8yGOSXTi7If4SVyRWg3hDRGupJjZtiWXzpLcXEgt3k/vGHd5ZOQDkr15681PrPh7T9fh8nVBcyQlSjRRXk0KOD1DKjAN+OaXSw1pK5l+Nrqey+Guo3FnPJbzx2gKSxOVZTxyCORTtYvZovGfhe2juZESf7SZYlkIEgEWRkd8H1rVg0Oxg0ubTik1xazgrJHd3MlxuBGCMyMxxjt0qtZ+EdFsb+3vobWR7u2BEVxPcyzSIpG3budiduCfl6DJIGab3uSlaKX9dDhNOl1G81TTluNZ1Fkv9X1C0kVbl1CwRl2VBg8HK/f+8AcAgAYnsL/U9W0XwLFJqt5Ab5547qWKUh5kSN+CT3IX73UHkEHmu5g8NaTbzW8sNpte2uJbmI+Y52yS53t15zuPB4GeMUW3hrSbOPTktrTYuls7Wg8xz5RYEN1POQx6560ltZ+X4D6v5/1+Rk6EbqDXfEej211M8VqsL2pu5nnaJpIzn53JYrkA4JPes/w5cz6Tqen2fieHWLbVrsSRm5lvWubO8kGSdo3ERkgbwNiYAIHGQeuTSbFLy9uRbqZb5VS5LEsJAoIAIPHQkcDmq1n4a0yyvI7qNLmaaIHymurya48rIwSgkdgpxxkYOOOlNb6g1poUdS1bULfUZYoLrZGpG1f+Efu7jHH/AD0RgrfgPaodduLifw7pkxKzXB1CAjfbyWoY+ZxlH3Mv459a6qq93ZW98ka3Ue9Y5VlQbiMMpyDx71Ek2rehcWlK4y0lvmDnUbe2gA+75Nw0ufXOUXH61yF/qbfYYdX0pdY2m6TbdzXAEMymTaR5RfhSDxhB2NdyQCMEZBrH/wCET0byhEbRmiV96RNPIUjbduyilsJz/dxxkdDUThJtWNITir3IPG//ACKVx/11h/8ARqVo61FfzaLcx6RKsV4yfunJxjnnnBwcZGaTUtEsdX4vxPImADGt1KiHByCVVgCc98Z6elSDTLb7A1m/nSQsc/vZ3dgc5BDsSwIIBBB47UOMnzeYlKKUfJ/5GVoV3FHqLWE0OpWd4YBI0F7cGdXHdkcsw4JxwR16emNDPquradd3ttb6s1+ZpBbvDdRpDEUYhUMZkAI453KScn2rrbPSbOxnaeFJGmZdhlnmeV9vXaGckgZ7DiopNA097iSdFngeVt0n2a6lhDt/eIRgCffrUypyaSv3/rboWqkU2/T+ty2iNcaeqXsQDSRYmjzkAkfMPfvXP/DqR5PAOmh2LLErwxsTndGkjKh/75ArQ8RRapcaT9i0TbHNdMIXuXYf6NGQd0mDyzAcAepGeM1d03T7fSdLttPsl2W9rEsUYJz8qjAro7s5uxneHrU211q7HXm1bzb1m8suG+xcD9zwTjHpx16eu3VDTdF07SJbyXTrZYHvpzcXBBJ3yHqeTx9BxV+l0Q+oUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBNb/xfhU1Q2/8X4VNUvcaCqj/AOsb6mrdVH/1jfU0IGJRRRVCCiiigAooooAKKKzvECXknhrUk0zd9sa1lEG04O/aduPfNJuyuNK7sZOu+MrKwv7Cw07UNPuL6bUIrWe285WkjRjhjtDZBHvWzqOt6Vo/l/2tqdnY+bny/tNwse/HXG4jPUVwdzq+k3Og+ErDRnWXyNQs/OjhGRbEcFZP7jZONp+Y8nHBI1dZ8SNb+K73TRdaVpPl20REl/atPLfht2FjRXQsFORgbiS2AB3p6K3m/wAkQnd38l+bNrxL4q0/w1oM2o3FzbM4geW2gkuFjNyVGdqE9c5HQHqKsaR4h0rXIx/ZmpWd3IEDyR29wkjR59cHiuN0u3ub74CXFrbxNLO9jdRxxJGVJIZwFC9R6Ade1dZoGvaTqumwrpeo2t5JFAjSRQTKzpx/EoOVPbmjZv8AruGrS/rsWRr2kHVP7MGq2Rv84+yC4Tzc4zjZnPTnpUuoarp2kQrNqt/a2MTttV7mZY1Y9cAsRzxXlmreJZdZ8L27/a9Jt3XUIX/si3s3e5sytwB8778IfVjGoycdxnr/ABUtpaeIrDVf+EistFv4IJIkGpIGgnjbqoyyfMGCnKtkDGRg0umv9aFdWdVFcQz2yXEE0csEiB0lRgVZSMggjgjHequn63pWriQ6VqdnfCLHmfZrhJNmc4ztJx0P5Vx2pXOqa78KDONPjt55JkNxDbwtJHNEJh5jqnysyOoLbepU9ec1Z0d7fWfFWn38XivQb6azgkQW2m24SR42A4b985CghTjHB+tPq1/X9f8ABFfQuw/ETw9Jr93p8mr6ZHBbwxyJdNfx7ZWYtlRzjK7Rnk/e7V1KsHUMhDKwyCDkEVxz6lZaF8SNTn1q8g0+C8sLcW8t1II0lKNJuAZsAkbl46812KsHUMhDKwyCDkEUdA62/rYWqNnrelahdy2thqdndXEOfMhguEd0wcHKg5HPFS6ikUul3UdyZBC8LrIYlLPtKnO0AEk46YBrifDepvZarpOh6dqek+JNPjjeNZrRQJ7FVU7WkKsykYwmcIST35oWrsD0Vzc8Q6rqukatpJgls3sb6+is3he2fzV3BiWEgkx/D021qaxqdvpWmyT3N7Z2TEFIZL2URxGTB2gkkenbnGa5Xx7rukW+o6BbXGqWUU9trEEs8T3CK0SbH+ZgTkDkcn1FdJqGvaRb6L9tuNVsorW4QiGd7hFjlJBwFYnB6HpUvSDfm/yRX20vJfm/+ATaJc3F5olpcXs1nPPJGGeWwYtA59UJ5IoOuaSNV/sw6nZi/wD+fT7Qnm9M/cznpz06Vzvw61K0vfh3ptrpt9az3ltYoskUcqs0TbeA4ByvI7+lR+CtS0W28N6fot3NBFq6kfarCb/j4NxnczmM/Mfm+fdjGPmzgZq2veaIi/dTOo1HWNM0hEbVtRtLFZCQhuZ1jDEdcbiM1bR1kjV42DowyrKcgj1FcDqV1caV8QtTub3VdL0mO4tYUtLnVLRnR4xnfGknmoAQxyV6nIPauj8G2EWneGYYLbUrfUoDJJJFPaoFi2s5bagDN8qkkDk9KS1Vxvexu0UlcmdD1Tcf9C7/APQ0X3/xFIZ1tFNUYUA9ceuf1rnvHqXEngm/W2R5OEMyR5LNCHUyAAdfk3cdxQwWpHL4ysp/Fek6To+oafei6aYXIhmWR4tiZH3W4545Fa95r2kadeR2moarZWtzIAUhnuER3ycDCk5OTxXLXOsabqvjnwsuizR3VtD9oBnt8NCpMPCBhwWwOQOgxnGRnO8QeImu4vE2m/adJ08xmSE6fNZPPdXn7sfvQqupwRjB2tgLknA4TdkEdWdV4p8Yad4Wt4zdXFqbmSSJRbSXKxvsdwpkwcnC8npj5TyK1NO1fTdXjeTSdQtb5IztdradZAp9CVJxXH635s3wl0q4KvJ5K2FxMyqWKojxs7EDsACT7A10k3iHT7rQb++0aa31oWsLsYbSZZd7BSQny5wT9Pwqpe7zX6N/oTG8mrdbFi017SL+9ks7HVbK5uo874IbhHdcHByoORg8Uuo63pWkNGuranZ2Jlz5YubhI9+OuNxGeorzv+3H1fVPCE66ppF5/pqHydNs3/0XdC+Y2l8xgOONpVScZx8pFbHiOaHRfEl7qun+I9MsNRms1iksNWjylxtyYyhDK46sDt3Ak9Mih6b/ANf18xrXb+tTtri5gtLV7m6mjggjXc8sjhVUepJ4Aqvbazpd5p8l9aalaT2cWfMuIp1aNMDJywOBgVx3iSe9udB8L6i9rHpVtHOk93Dc2zTR2f7pthkQFDtVsc8bTgnGK0PDCQ3nia/1eHxJpGqyS20cM8OlxBQCrMVd/wB65zgsO3H0otq0F9EybSfiBoOp6he2p1TToTBdCC3Jvoz9qBVSGUcd2K8Z5FWPEuq6jY3+jWWlPaxSajctC0tzC0oQCNnyFV15+XHWszQ9UsNH8U+IrPVr23sbi71FJreO5lEZnRokUFM43fMpHGeRirXin/kafCX/AGEJP/SeShapfL9A/m+f6nSRGSK0U3ssbSKmZZEQohOOSAScD6k/Wq1hrelapHLJpmp2d4kP+ta3uEkCfUgnHQ9areKbOy1DwvfWmqXD21pMgWWZBnyxkfMeCMA8kngDJOBk1i+GNburzXzYNeaXrttFZhhqumpt8tgQPLkAZly33htYfdPy9KFqweiRPafETw7cazf2Umr6ZDDaiMxXLX8e2fcCTt5x8uMcE9e1b9/qlhpVuLjVL62soWYKJbiZY1JPOMsQM8GuYh1Sw0L4ga82t3tvpyXkVs9s91KIlmCqyttLYBIOMjqMj1qXxP4hOl+IdOtRPpmmCa3lf+1NTjLRqAVzEvzJ8zcN97ovQ8YXRD73Okj1Kxl07+0Iry3ey2F/tKyqY9o6tuzjAx1qKLWtLn1FtPg1KzkvVXcbZJ1MgGAc7Qc4wQfxrz7TZHk+HnjvzGRv9IuyDHbNbqcwIxIjYkqCSTyTnOe9djpGgaX/AGDoarZRxrYIk9sI8p5blMFuMZzubOc5zk0/+B+Iv+D+BVl+IGgweJjpUuqaaka25le5a+jCrIH2+UR2bv1zx0roRe2psftouYTaeX5v2jzB5ezGd27pjHOelcvql7a6N8Sbe/1a4isrObS3gS5uHCRmQShtm48A45x359K2tautMl8JX11er9t0trN5JRAd/nRbCTtIPOR0IP41N/cv6/mxrWdvT8kaME8Vzbxz20qTQyKHSSNgyup5BBHBFVLfXNJu9RksLXU7Oa9iz5ltHcI0iYODlQcjBrMtvLuvhsv/AAisMlqkumf6BG4w0eY/kByTz05yfqa5PSPs+pafoOmnxNoNnLYzwyR2C2Jhu4pEPzRlXnJDH5lPy85PrV295r+v6RN/dud/e65pOm7/AO0dTs7Tyyqv59wibSwJUHJ4yAceuDVr7TB9l+0+dH9n2eZ5u4bduM7s9MY5zXKaXaW8nxW8RXMkKNPFZWiJIy5Kqwk3AHtnA/KsCzbZ8NvDctzzptvqga93fcSFZZApb/YVthPYY9BUr+vvKe/9drnVWHi211bxl/Zek3tjfWa2BuHltpRIyyeYF2kqSAMHOMZp13qer2HjHS7GaaymsdRaYKq2rpLEETcPn8whv++RWfYana6r8UxPp7+fbLo7ItwnMcp85SdjdGAyBkcZyM5Bxb1//kfPCn+/df8Aoqmtk/X82S938vyR1FFFFIoKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAJrf+L8KmqG3/AIvwqape40FVH/1jfU1bqo/+sb6mhAxKKKKoQUUUUAFFFFABRRUc8y28LSyByqjJEcbO34KoJP4CjYCSiuO0/UYdX1zUHvZNW2x3Cw20UMdzEiKFH3ggABJJJ39sdq3rnWkhvZLS1s7q+mhUNMtuE/dA9AS7KMn0GT+lZxqJx5jSVNp2NOisafxRp0GkW2pEzPb3EywjZGSysSRgr14IIIGTU1lrcd3qJsZbS5s7nyROqXAX50zjI2seh6g4PPSq54t2TJ5JJXsadFFYmp+KLbTNag0n7JeXV9cQNNDFbRBt4DAEbiQF65yxA984FV5Em3RWHbeLdOm0e71C4W4tBZTm2uIJo8yxyggBMIWDE7lxtJzuFLaeJkm1KOwv9MvtMuZ4zJbrdiMicLywUxuw3Ac4JB/WgDbrMvoNdkui2m6jp9vBgYS4sHmbPf5hMg/Sk0zxBYappdpewyeWLyEzxQykCUqPvfLnnGRnGeoqfSNTh1rRrTU7VJEhu4lljWUAMFIyMgE07AR6fDrMczHVb+xuYtvyrbWLwsD6ktK+R7YH1rRprsUjZlRnIBIVcZb2GeKxdCuUn1XVRnUUnSRPNt7yRGWLK5Aj2kgDB9aly1SGo3TZuUViS+JoUvr20t7G9uprHBnEKLhQVDZyzAHr068HitSzu4b+yhurV98MyB0bHUGlGSlsNxlHcnoprukUbSSMERQWZmOAAO5rhvEPii4vbPR5tKs9SgsrnVLQJqG5I45o2kHG3f5m1h6oAR7VW7S9PxJ6XO7orO1HVZrGQR22kX+ott3v9lEYCDtzI6Ang8KSR3xkZzdQ8TNL4LfWdDs7u5ElvIyFFjRoCqnl1kZfusMEDJ46Gk3ZNgldpHR0VgeFdav9W0ixe/0u9gaS0jka7maDZKxUZICSFhnOeVH4dKdf+KE0yQveaVqKWKyiJr/ZH5Sknbkrv8zbnjOzHfpzVtWdiU01c3aKytV16LTLy1so7S5vr26DtFbWwTcUXG5iXZVAGQOTnkYBqZ9Xgt9FfU9QSWxhiQtKk6/PHjgghc59tpIPGM5qelyi/RWJY+JUub+3s7zTL/TJbpC9t9sWPE2BkgbHbawHOG2n8jjGPinWF8fzaeNE1KS1XTxKLVWtdxbzSvm7jKPlIGMbs/7I60+thdGztKKZE7SQo7xtEzKCY3I3KfQ4JGfoSKJZUgheaVtscalmY9gBkmk9NxrUfRWNpHiL+2fJlh0q/hs7hN9veSrH5cq9QcK5dcjkblX04JArM8YxNa32i31tdXkM0mqW8DrHeSrG6EnIMYbYc+uM0+qQr6NnWUVV1CeW3sneC0uLx+gitmRX57guyrx9ay/BUsE/hGzktbnUbmI78S6pKJLgkOwIdhwcHIGOwFAzerE8QaHdatdaZdWF7DaXGnXDTIZrczI+UZMEB1PRs9aS18Tx39yw0/Tb+6s1mMLX8aJ5IYHa2AXDsAeCVUjg88GptR19LLUF0+1sbvUr0x+a0FoEBjjzgMzSMqjJBAGcnBwODSDuaUAmW3jF06STBRveNCise5Ckkge2T9akqlpOq22s6ct5Z+YELMjJIm143UlWRh2III/lkVX8P+ILTxJYzXdjHPHHFcSW7CePYSyHBIGen+eKfURq0UUUhhRRRQAUUUUAFFFFABRRRQAVgz6FqF54qsdUu9RtjbWDSmG3itGVyHXb8zmQg49lFb1FABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUATW/8X4VNUNv/ABfhU1S9xoKqP/rG+pq3VR/9Y31NCBiUUUVQgooooAKKKKACiiigDH0OxubO+1eS5j2Lc3hliO4Hcu1RnjpyD1qBIb/SNa1KeDT5L+3vmWVDDIitGwUKVYOw4OMgjP0rfoqORWSXQvnd22tzk5PD9/HounwhVluBqq3twEcbUBcs2CcZAz+PpXWUUU4xUVZf1ol+gpSctwrAl0y7b4i22qLFmzj0uW3aXcOJDKjAYznoDzjFb9FV1T/raxPSxieK9LutV0VF0/Y11a3MN3FHIxVZGjcPsJ7ZxjPY4qmbfUde8Q6Ve3emT6XbaY0kxW4liZ5pGQoABGzDaAxJJIPTiunooWgPU8ufwp4jg027ntbBWvdNuJYdLi+0IPPt5Hl3HOflysq8E5zEK9F0myGm6LZWKjAtrdIgB/sqB/Spnu7aO6jtpLiJJ5QTHEzgM4HXA6mpqa0X9f13+Ynq7jJXMcTusbSsqkhEIy3sMkDP1IrntIGoQ+I9UubjSLqKG/kiKO0kJ2BUCncBIT1HbNdJRUON2ncuMrJq25hWOnXUOpa/LJFhLyRTAdw+cCIL68c+tWfDdnPYeGdPtLtPLnhgVHXIOCPccVb1DULXS7CW91CdYLeFdzyN2/xJPAA5JOBVgHcoI6EZ5GKIwUdUOU3JWZU1ayOpaNe2KyGI3Nu8IkH8O5SM/rXHXOn+ItR0TQtNOjfZv7Mu7N7mSS4jYSiJ1BMWGJ24Bb5tp6AKSTjvaKtaO/p+GxD1/H8Tj9ZsdYuPFMrzWupXumG3RbRdP1L7IsUmTvMu2RGI+7gjfgZwMmpPDHh++s/hzJouoIttdSC6Qgy+aF8yRyp3dW4YHJ59ea6yq99e2+m2M17fSrDbwIXkkboqjqalpcrTHre5keF5NQt9Ls9L1HR7m0eztlia4aWF4pCgC/Ltcvz1GVHHXBrjtU8O+JNV0C9tbzT9SudVeZmNy+r7LR0EmV8uJZMfdA+Vo1HUk56+mxSxzwpNBIskUihkdDkMDyCD3FPqm3zXe5CS5bI57xNpsN81m9xoV1qbW7FoprG6WCeBsYOGMkZAI64b0yO9Zn/CNa1qXw5uNI1K8ke8eTfbNdTZkRFkDxpLInVsKAzLnrwTjNdpRSK7HIaRpEUuq2dxe+H9ct7i2zJHNfav9piicqQcA3Dk5BIB2d+1WNQtNQsPG665aadNqVvLp/2N4raSNZI2EhcNiRlBByRwcjHSunop31uK2lv67jInaSFHeNomZQTG5G5T6HBIz9CRSTBmt5AkaSMVICOcKxx0JwePwNUL/QbTUbnz7ibUEfaFxb6lcQLj/djcDPvil0/Q7TTLgzW0t+7Mu0i51Gedcf7sjsAeOuM0tx7HMaRod5Z6/Yvoem6loOnxNKby1ubxJLaVWycRxLI+xt+DkBABu65xV3xguqXs+nQ6fod5dpZ6hBdvNHLbqjKpJIG+UNnnuAPeusop9vIVt/MoS6hcppiXS6PeyTMcGzV4fNT6kyBPyY9axPAsGo2Ph2LS9U0m6sZIfMYyySQsjbpGYAbJGOcN3AFdVRSA5Xw1Hq+gabbaDPpEs6WrGOO/jni8l485DMC3mBsHBAQjI4ODmq2s+H7hfF0+rrZ6hf213bRwtHp2pNayxOhbBI82NWUhvXII4HJNdnRT31GZPhzTrbTdLZbTT7rT/OleaSG7n86UuTyzPvfJOM/eNL4fuNauLGZvEVnBaXIuJFjSB9waIH5GPJ5I/wD1DpWrRR1EFFFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUATW/8AF+FTVDb/AMX4VNUvcaCqj/6xvqat1Uf/AFjfU0IGJRRRVCCiiigAooooAKKKKACiiigAooooAKwbzU9TbxUukaetqiGzFw086sxT5yv3QRuzxxkY5OT0rerlrpLx/iLiwnggf+yhuaaAygjzfQOv860ha+pMvh+78x83iS7g8LarePFAb7TJGhcAHy3YY5AzkAhhxnir1tNrwvrU3UFo9rOG81YchrY4yuWLfvM9OFHWopvDKSeGb3SxdN5t6Wee5ZAS0jEEttGOOMAdhitxRtUD0GKblG2grP8Ar8Dn7yaeDxnpcVxHYzpcCbypPsxE0AVc4Dlj1zzgCpdX1TULfXNO07TY7cm8SUtJPn93twd2ARnqeOM8cikvtF1O71q11BNStIzZmTyUNkzcOMHcfNGTjuMVbm0prjWtP1GScBrOORGRY+HLgDPXjp05p3j7t+z/AFt+ga6/12K2m6hqF+NRs2kto72xnEXniFjG4Khgdm/IODjG739qzvBUepf2XuN3a/Zxdz74/srbyfMbOG8zA59j+PWtzT9L+wahqN153mfbpll27cbMIFxnPPTPaq+m6Rd6VM8drewmweZ5vJktyZF3HJAcOBjJJ5U0cys0vL7+omn+L+7Uz9VUap4+0rTJzm2s7d9RaLHDyBgkefULlm+oB7VY8Va3qGj/ANlJpNtb3M19fLalLhyigFGOdwzjBUdjxniq/iYS6RrOn+JYopZoLVHtr6OJSzCByD5gUcnYygkehY9q09X0caxcaVOLjyhYXa3QGzd5mEZdvUY+9nPPSsVsvXX7/wDI06v00+7/ADKkeo6xpK6jN4jW1msbW1+0peWkZiDYBLoY2dzkAAg5wc1QuNf1/TdKtNd1JdOOnzSRedaRI4lt0kYKD5pba5UsuRsXPODxz1N3axX1jPaXK74Z42jkX1Vhgj8jXOx+FL+WztdN1XWVvNLtJEdIxabJpRGwMaySbyGAwM7UUnA9wWt9fL/gie2n9dhG8ZJp95qcesR7VttSis4PIQklJI1cO2T0GXJI6BenHOjoWqT6u2rfaI4hDbX8lrDsB+dFCgls9TuLDj0qhf8AguLUfEV/f3V2Wtb2zNu9n5QwHKlPN35znYSMY7mrGkWSeDfCTjVL9rv7P5txc3Zi2mRmYux2gnnnGBS6a9v6/L8R6307/wBfn+BU8Gf6Bea5oCH/AEfTLwG2X/nnDKgkVB7KSwHtiuqrnfCFjdRW19qupwmC81a5Ny0DdYU2hY0PuFUZ9ya6Kn2v2X5CXkFFFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAE1v/F+FTVDb/xfhU1S9xoKqP8A6xvqat1Uf/WN9TQgYlFFFUIKKKKACiiigAooooAKKKKACiiigAoormbyOa+8eCye9uorMacJXhgmaMO3mEA5BBH4EE4A6cVUVd2E3ZXOmorirm/vIfBPiAJdTl7C4kht7gufM2grj5upIyRnrxW7baNPBeWt2mq3blQ32lJpC63GRxhc7UweflA9KtwSV2xc3QuTah5OqW9l9kupPPVj58ceYo8DozZ4J7VcrnL5GtPG2j+RcXIS7FwZo2uZGjbagI+QnaMZ7AUmuC4uPFej2Ud7cW1vNHOZlgkK+YAFIGR0+o5HOCKXKna3VP8AC/8AkK9r/wBdjpKr2V9b6jbmezk8yMOyFtpHzKcEc+4rE0VGuZ9a0u4nuZLe1ulWJjcyCRVKK2PMB3YyT39qreCtMgGl/aPMut6Xc4A+1y7DiRhym7afxHPWnyJJt+X4hzPobOu60mi2KSCFrq6uJBBa2sZAaeQ9Fyeg6knsAT2q7cXAtbOW4lUkRRl2VeTwMkDOK566xN8UdPScZWDS5pYATxvMiKxA9QvGfRjXTMqupVwGVhggjIIrHVxut9f8i/tW/ruVtPv49S0yG+gjkEc0YdUcANj88frSabqMOq6fHd24ZUfI2uAGUg4IIGeQRViKKOCFYoI1jjQYVEXAUegArFsf+JX4nurE5EF+DdwegfpIv8m/E0m2pJC6G7RXNeTLqHjO/tpr27S1ht4nEEM7Rgsd3OVII+gIz3zirnhS7nvPDdvLdSNLIC6F26sFcqM+pwBzSjO7t6/g7D6lvSNVh1mw+120csab2TEoAOVOD0JqvZ66lx4gvNHuLeS1urdRLEJGBFxCePMXHo2QR1HHrWlDBDbR+XbxJEmSdqKFGTyTgVzXij9z4n8K3EP+vN88Bx/FG0Llx9PlU/hVRvon6fP/AIf8BPZnU0lLWdrssUOi3Lz+eFCE5g37gQMjlOQOOT09aG7K5SV3Ylg1WzuXt0hm3NcxmWIbSNyjGTyOOo61crhdBis/7Z02Mf2gDHp6Hn7QAHLD8Any/wC6feu6qiQorJ1m6hWa1tH+2SzTlikFnJ5bOAOSWyuAP94Z96x7bVrux0zxBgXGbE7oEunEjx7lzgsCcgHnqeKm+5Vrux0d7qMVhLaxzK7G6mEKbQOGIJ556cVZZgiFj0AzXJ6jaR29x4fla5nnnlvEZmluGYN8pJIQnaOT2AxVtCdVvtXN7cTxw2T+VHHFO0IUbASzFSM59+MCh3Sflf8AT/MFZ2/rubOm38WqadDe26uscy7lDgAjnHOM1arktNi1Cb4eWK6Sx87aCyrJsZ03HKq38JPrWrot3A11dWarfQTxbWe3vZDIQD/ErFmyD0+926DvT3aJ6XNiiisTU7iGbVFslTULmZIvMaGzm8kICcBmbcmSfTJ6dKQzborF8KXc93of+lPI8kM0kW6UguQrEDcRwTjvW1TAKK4ZZLtfB95qzaheNdQTP5OZ22oFlxgrnDd/vZ447Vrapd3E/iG2sBFdyQC1Nw0dnMsTSNu28sWU4HoD1IpX2DY6OiuSuX1Wy0G5kuPtUCW95HJAZZlaRody5VipOep6k5FJNrt1f3sNop8gtqEZhaJiDLBl8k+ufLPtgij+vyA66mSSJDE0krBEQFmYngAdTT6w/F7EeHJV6JJJGkhzjCFwD+nH40MDQ0u/OpWQuvIaGORiYt55dOzY7Z9KuVU1Gc2Gj3M9ugLW8DOiAccLwK5y0GqPb6fdWcGpSTu6PcSz3UZilRuWwnmEKOeMKCMUdbB0uddRXDX9xfNql7s1C6jVdTht41SQgIrqN3HQ9sA8DqOpp8l3e/2OYYb24jaPXPsyS+YWcJu6EnO7r3zSTv8A16f5g9P69f8AI7aq9ve291PcQwSbpLdwko2kbSRnHPXj0rHjiaz8Spp0dxdy29zZs0iyTu5RgQAwYnK5yRwRVXQ9Lt31rWgZLvEVyoXF5KM/IDzhvm/HNNav+u9gf9fcdVUc7vHbyPDGZZFUlYwwG844GT0zWLfXb6T4ljnuJ5PsN3AyFWYlY5EG7IHQZXP1Iqz4dFy+l/a715DLeOZ9juSIlb7qgHoAMUbhsWdL1KLVbFbmEMhyUkjf70bjgqfcVcrB0v8Ad+Ltahj/ANUwhlYejlSD+YANb1AeRT1LUP7NtWuGtZ7iONWeQwlPkUDJJ3MP0zU9tOt1aQ3EYISZFdQ3UAjPNU9f/wCRb1L/AK9ZP/QTTtJUP4fslOQGtYwcEg/cHcdKF1/ruHYltr77Td3MH2W5h+zsB5kse1JM90PcVarB0PfFr+tWvnTyQwtD5azTNJtymTgsSetVI4Zr/WNeE99eLFbFRDFFOyBCY854IP4dOvFK9lfyuNK7OporkH1a9uNA0FSZ2e/IE727KkjgKTgEkAE465HfFamkJfw6rcK9vdxae8QZBeTrKyyZwQCHY4IweT1BqutielzboorC1KeG51R7OOPUrmaGIM6Wc/krFnoSd6ZJ9Mnp0HOUM3CQASeAOtZ+n6wNT2yWtlcm1ckJctsCMBkZxu3YOP7tZmkSXeqeDY5Zr24ilUSKzqELOFLKA2VI6Acj86d4StJD4asXe+uHjeDHkfIqrn0IUN/49R1YG1De2893cW0Um6a2KiVdpG3cMjnvx6VYrldK0q3fxNrcZku8RtBtIvJgTlM8kNlvxzU+q3VxN4lj08RXklvHbeeY7SZYmdixXli6nA9Aep5ovsB0dFcvNJqlp4R1c3X2iBog5tXklVpQmOMspPIOec5q9Y6VOFsLxdTumkChrgSuXWYFem3O1cHoQPzo/r7wNqiiuf126uDrVhp8aXTQyo8si2sqxvJtxhdxZcDnJwc0AdBRWFp0l3Yw6k+pC5trCIeZA9zKssiLglhkFicHpnJ7Vm/bJYNQ0ee1i1SGK7mEbtd3AdJVZc52b22njPAXv9KOtv61DodfVS+1GLT2tRMrt9pnWBNgBwxzgnnpxWYu/VfEt/bXFxPHb2SRhIYZWi3lxkuSpBPTGM4qHxNbstto1ulxKG/tCJRMxDOOG5yRyfc/jmjt8gOkorB08yWniu609JppLb7IkwWaVpCrbiDgsSce1b1HS4BRXO3iTXfjRLM3lzDbfYfMeKGQpvIkx1HI+owe2cUagt5o+kpbpfyyfab1Yknfl4I3bpuYnJHIBPr7Uf1+NgOiorn5Y30zxFp9va3N08N6kiyxzTNLjauQ4LElTk4OOOlZVl4lurTTbWK4PmqkUiXM8jEusmZAgz7+WevqKTdhpXO1orP0ETDw/Y/aZHlmaBWd3bLEkZ5P41oVTVnYlO6uFFFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBNb/wAX4VNUNv8AxfhU1S9xoKqP/rG+pq3VR/8AWN9TQgYlFFFUIKKKKACiiigAooooAKKKKACiiigArnrvQ5L3xn9snWdbQWAiEkNy0R3+ZnHyMGPH4V0NFVGTi7oTV1Yoto2nvo7aWbZRZMu0xKSuRnJ5HOSeSc5NXQMAAdBS0Um29wsZVx4b066vFu5/tZnUsUcX067N3Xbh/lB9BirbadbPd2108Zae1RkidnYlQwAPfnOByc1aop80u4WRXt7G3tbi5ngj2SXTh5m3E7mAAB56cAdKrxaJYwXrXVuk0MjyGR1juJFjZiOSYw20k/StCijmYWRg+JNKvbiSz1XRBGdU05yY45W2rcRsMPEW7ZwCD2ZRW6CSoJGDjkelLRU9BhVDU9M+3vaSxTeRPaTCWOTZu46MuMjgg4q/RQ0nuBXjsbeK+lvEjxcTKqSPuPIXpx070WVjb6darbWcflxKSQu4nknJ5PuasUUWQBWBDpd5feL21bU1EVtYK0GnwBgxYsBvmbHQn7oHYAnvW/RR1uAVHcQR3VtJbzruilQo65IyCMEcVJRQBWgsLa2nM0MW2QxrETkn5Vzgfqas0UUwKd7pdpqEkUlzG3mQkmOSORo3XIwcMpBptro9hZLOttbKq3GPOBJYScY5z19/XvV6ikBmReHtMheF1t2ZoGVomeZ3MeM4AJJwOenT2qSfRrC5umuJoC0jhRIBIwWTHTcoOGx7g1fooApLpFkunx2SRMkERzGFkYMh9QwO4Hk9+9Ps9OtrFpGt0bfKRvkkkaR2x0yzEnA9M1aooAKoXWjWN5dfaZo3ExTyy8czxll/unaRkexq/RQBWsbC1023MFjCsMRYvsUnAJOTj0+lWaKKYFD+xrD+zJNP8j/RZSWePe3JJyec561Jd6ZaXxia4jO+H/VyRuyOnGDhlIIHtmrdFICmdKtX0+WylEs0EwIdZp3kJ/4ExJH4GmJomnR3FpOlsqyWUflwMGPyLjGOvP45q/RQAVW1Cyi1LT57O4/1cyFSfT0I9weas0UbhsUdKW9GmiHVlRpoyYy6nIlUdGx2yOopsGiWVrIrWwniVW3LGl1KIxznATdtx7YxWhRT8wKD6Lp8kryPb5aSdbhjvbmRfunr29OlZ+s6EJbOGHToGAfUUuZ9spU9fnYEnI/D8K36KQFS0021sZHkgRjLJ9+WWRpHI7DcxJx7ZxTJdIs5bxrsrLHO4AdoZ3i346bgpAb8avUUAYeu2E2uNFpzWhW0WVZZbmRlwQOdqAEnJzjJA4z1rYmZ47eRoYvNdVJSMEDcccDJ4FSUUdLB1MzRNOlsreWa9YPe3cnm3DL0B6BR7AcVp0UUwKl9p1vqUfl3fnFMFSsc7xhgeoIUjP40ltplvZ2rW8BnETLtw1xIxUYxhSWJX8MVcopAZtvoNja3bXMH2kTMQXY3crbyOBuBbDfjVmPT7aKa5ljiw90QZjuPzYGPXjj0qzRQBROj2B02PTzbg20WPLXccoR0IbOQffOaltbCGzLGF7ht3XzrmSX8tzHH4VZopgFULnRbC7u2uZom810COUldPMUdmCkBh9c1fopAVrTT7WxsfsdpCI7f5sICcDJJP8zT7S0hsbSO1tU8uGJdqLknA+p5qaigCjPpFncXZumWWOdlCs8M7xFgOmdpGfxp95plrfPG9xG3mRZ2SxyNG65GCAykHHtmrdFAFM6ZbNp8tlKJZYJQQ4lndyQe24kn9atRosUaxoMKoCgegFOooAKq3un2uoKguo9xjbdG6sUdD6hlII/A1aooAqR6bbR20tuRLNFKMOs8zy5GMY+cmq0Xh3TIXgdYHZrdgYTJPI5jx0C7mOB7dOnoK1KKPMCldaVZ3lwtxKjrOq7PNhleJivoShBI9jTn0y0kito3hBS1kEsIDEbWHQ9eep61booAriyt11Br0R/6Q8YiZ9x5UHIGOnWrFFFAFf7Fb/2h9u8v/SfK8rfuP3M5xjp1p11aQX1rJbXcSywyDDIw4NTUUAU7XSrSznM8KO0xXZ5k0rysF9AWJIHsKrv4c0l7a6t2s1MV5J5s43t8zZznOcjn0rUooAaiLHGqIMKoAA9BTqKKYBRRRSAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAmt/4vwqaobf8Ai/Cpql7jQVUf/WN9TVuqj/6xvqaEDEoooqhBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBNb/AMX4VNUNv/F+FTVL3Ggqo/8ArG+pq3VR/wDWN9TQgYlFFFUIKKKKACiiigApDwOmfalprEqhIUsQMhRjJ9uaAMPSbpbjxHqQcalBcCONntrqVGijBBAKBGIGcc81U8S6yBqGn6cj38MctyRctBbzKzqqltqMq5OTjOw5x7ZqSx/tFPFd7fS6NdxwXccMasZICU25yWAkPHPbJ9qu6vY3N1rWjTwR7orWd3mbcBtBQgdevJ7VhZuCS79vM6PdU2327+X+Zc0xYFsQbX7V5bEkfazKX/8AIvzCrlFFbrY527s5NfiFp76a+pR6dqbafBKYrm78lAkBD7CTlgWA6koGwOvORWtqGvx2d+lja2V1qV40fmtBaBAUjyQGZnZVAJGAM5POBwcVvBWmXekeF4rTUIvJnW4uHKbg3DTOynIJHIINQXFtqOkeMLvVrPTpdTtb+2iikjt5Y1lheMvg4kZVKkP2OQR05o7XDv8A11/yLEXi7T5rSznWO5U3N6LBomjw9vNg5WQZ4xjGRnqMZBzVu61+xtZbONX+0G7vDZKYCHCSAMSG54xtOe49K5i88O6uPDWoXcNqsur3GqR6pHZrOoClGjxHvOBnYmCemSeo5pmleFNTsPFmnM9uh05Ab+4mEg+W8aIxuNvU7t27OOufWmvP+tF/wfwE/L+tX/wPxOiXxNHPqFxb2Om395FazeRcXUKx+XE4wSMM4dsAjO1W9s1t1wuuaHez6hdzaBpF/perTSqRqdpfIlrJtPyvLHvy/B5BiJ6gEjmu5XIUbjk45IHWhbB1EdxHGztuIUEnapY/gByaxY/EyjV7Wwv9K1DT2vC6201wsZjlZRu25R2KkqCQGC9D34q/rCX8miXqaNIkeoNA4tnk+6smPlJyD39q4e28ParPq3hy6OmapD9huA97JqWsG4JPlOpZI/Mdcbj1G088LgnCW/8AX9aA9EdHP4xt01TUdPs9L1K/udNKm4W3jQBVKBgwZ3VT1xjO7g8Y5q1L4n09dLsL22Ml3/aQH2KGBf3lwSu7ADEAYAJJYgDuRVXS9LvLfW/E1xNDtivpY2t23A7wIVU9+OQRzisXTtB1fStF8IXP2F7i50eBobqxjljDkPGFJUlghKkD+IcE80LbXy/4P9dAe+nmX/Dmp3t/468QR3kF5aRxW9oY7S6lRvLJEmSAjsozgdDnjmutrl/D9hqq+L9a1XU7NbWG+gthAgkVyuzzMq2D94ZBOOOcAtjJ6imPq/66GTqWvxWGoRafb2l1qF/JGZfs1qE3LGDjezOyqBnjk5J6A4NPudct7LR1v72G4g3MI1tmjzKZCdojCrkEk8Agkd845rNu7XUdL8Xz6zZ2Eup215ax28sMEkayxNGXIYCRlUqd5z82QccHs7X7LUtU0vTr61swt9YXaXi2UkwBkADAoWHyhtrepUNxkjml0X9df8g6v+un+ZbsPEUd3qp0y8sLzTL0xGaOK7EZ81AcEq0bspwSMjORkcVleU1j8ULSG3urwwXWn3E0sMt5LJGXEkeCEZiFxk9AOtWIbXUNY8WWWrXmnzaZbafBLHHFcSxtJK8m3JxGzKFAXu2ST0GMmleSau3ju11WPw1qMlrbWc1sSJrUFyzoQwBm+7hT1wenFNfEvn+T/wCAJ7P5fodE+sW6eIotFKSfaZLVroNgbNisFIznOcsO1Z11400yy0y+v7lLlYLG+FjMRGGO8sq7gAclfnHv14qLVrPULXxlZ67Z6fLqES2MtpJBBJGsiFnVw37xlBHykHnPTg1h/wDCOa9N4a1GG6solvLnX474RRTqy+UJY2JDHHQKeoBOOnSlHW1/697/ACCWi0/rT/M27bx1Y3F/DaPY31vI92bKQyom2GbaWVGIc5LAZG3djIzjNLN44tFm1KO20zUrv+y5GS8eGJAsQVQ27LON3B6LluOQMjOM/hrVjqTSi0+Q+KV1AHzE/wBQIQu/r68Y6+1aVlomoRab4uikt8PqN3PJajev7xWhVVPXjkEc4pXfLfy/G0f839xVvet/W7/yRrX/AIjs7K3sZIY5r6TUSBZwWwXdP8u7ILFVAC5OWI/Pirthdve23mS2dxZSBirQ3AXcv4qWUjHcE/mCK5y50UP4Q0Wy1PQZtTe0ih3JbXCRzW8qKMMjl0xggglXB6dQTi74RstUsbG6TVJblonuWezhvJhNPBEcfK8gJ3Hdux8zYGBk1bSTaITbSZa0zxDaarrOq6bbR3CzaXIkczSR7VYsu4bTnn9Py5rWrJ0y41uXWdVj1Wzt4LCKRBp8sb5aZSvzFhk4wfYfj1OtS6IfcKKKKQwooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAmt/4vwqaobf+L8KmqXuNBVR/9Y31NW6qP/rG+poQMSiiiqEFFFFABRRRQAUUUUAFFFFABRRRQAUUVg3mp6m3ipdI09bVENmLhp51ZinzlfugjdnjjIxycnpVRV3YTdlc3qK5mbxJdweFtVvHigN9pkjQuAD5bsMcgZyAQw4zxV62m14X1qbqC0e1nDeasOQ1scZXLFv3menCjrVcjFzGk93bR3UdtJcRJPKCY4mcBnA64HU1SvtAs9RujPcTagjkAYt9SuIF4/2UcD8cVRvJp4PGelxXEdjOlwJvKk+zETQBVzgOWPXPOAKl1fVNQt9c07TtNjtybxJS0k+f3e3B3YBGep44zxyKXJe1uv8Awf8AIObct6folrpkzS20t87Mu0i51CedcewkdgD7jmtGsTTdQ1C/Go2bSW0d7YziLzxCxjcFQwOzfkHBxjd7+1Z3gqPUv7L3G7tfs4u598f2Vt5PmNnDeZgc+x/HrT5NLti5uyOj1DULXS7CW91CdYLeFdzyN2/xJPAA5JOBUzSokJlkYJGq7mZuAB6nPSua1VRqnj7StMnObazt31FoscPIGCR59QuWb6gHtXSXEK3NtLBJnZKhRsdcEYrJ35br+v6Zel7EM0dtq2nlFmd7eYAiS1uGjJGc5V0II/A1m/8ACJad/wA/Osf+Du8/+O1LeWMWm+DbmytyxigsnRS2MkBD1xXJRyvaeFhpE5Y+W9rc25P8UTyIcf8AAWyPyrN1LStbt/X4C1smz0JFEcaouSFGBuYk/iTyadWZbalNN4kvtPZUEVvDG6MAdxLZznn2rITxLqEuj6ZcRQ2xuL68a2IIYKoy4Ddc8bQffnpVOovv/wA7fmG39fM6qo4J4bmIS20qTRkkB42DA4ODyPeuZ0nxJqF3fWMd3FbCK7eeIeWrBt0X8XJOAem38c9qo6Jf6npvg030K2ptLWSRmicMXlXzG3EMDhe+Bg9O2aj2yuu1r/l/mM7iiuf1TxElvqcFjBeWNnvh897i9b5Qp4Cqu5cknnrwB3qKLxPJL4Z1S+i+zS3Gnl13xEtFKQMhhz0IPTJx61TqxV/K/wCG4bux0tFYtpc67JJZTTwWj21wMyxxZDW4IyDuLYf3AUde9bVaJiTuFFZWoahcjVbfTNO8kXEsbTPLMCyxoCB90EFiScdRUdlqd/Jc39hPFbSXtpsZGV2jjlRs4PRipGDkc1HOrjL0+q6fa3K291f20M7Y2xSTKrHPTgnNW65TwxJqLXmqubS0O/UXE0hum3LgAYA8v5gB0yR+FdXRTk5RUu4uoUVk+INUutLt7RrKGKaSe7SDZKxUENnuOnTrg/SiK61SzmnOrrbyWkdv532i3UptYfeQqWJPHOeKOdXfl/lcfWxrUVzUmtarb6LFrc6Wn2Ntsj2qq3mJG3APmbsEjIONo7jPepZvEyWF3qS6in7m1liSLyVJZldc5OT2wT9BRzpOz0EdBRXPya9drZ61cQ2q3AsJvKhSMHL4VSxPXoSeg6CrOh6k+pIzjUdPvo9itm0Uo0ZPZlLN+uDx09BTTdl6ga9Rz3ENrA011LHDEv3pJGCqPqTVNptTGvpEltEdMMBZ5y3ziTPTGemPb8al1PTodW0yexuS4imXaxQ4I5zxmm23FuO4+paVg6hlIZSMgg8EUtMjQRRLGv3VUKPoKfViCq93f2mnxCW/uobWMnaHmkCAn0yasVXv/wDkHXP/AFyb+RpDSu7EsM0VxCk0EiSxONyOjAqw9QR1p9YHh66jsfAFhdTf6uCxWRsegXNZsHjFh9hnmv8ASZlu5kjaytpMzQBzgHduO4jIyNo7+lbezbk0uhmprlUn1OxorAfUtWuvEl9plgLOKK1iik8+ZGc/MD8u0MM5x1yMY6HPETeJpP8AhXt1r5iVJoLWaQoOV3puH5Er+VZyi4x5mWtZcpt2+oWt3dXNtbTLLLasFnVefLYjIBPTOOcdRkeoqS4urezh867njgiyF3yuFXJOAMn1JArM8OWEWi+F7aMuXYRefcTN96WRhudz7kkmuP8AEmoavrngWDV2ks7fTbq4tZY7T7OzTBDOm0mXzAuTwSNnGcZPWla0regJ3Vz0iisTxDqraYsP/E50jSEkJxLqfzbyP4VXzE+udx+nes/Stb1DxN4Oe7sbnT4bqOWWCWVYmuIJdhKlkAdThuCDk4Bxz1qW9G+xSWqR1dFcb8NotWXwXoj3F7ZvZGyTZClm6yLxxmQykH/vkZ9qPFXiybQDdyprWgRPap5g0y5Y/aJ1HOA3mDazDoNjc465qpWi7Ex95XOyormPEniiXS7PSpoGgtLfUHAk1C7jZ4bUFcrvClfvEhQSygdSe1WX8QLp/g651u8urPUIreJ5RPYHEc4HTblmwc8feb69gnpe/QFra3U3qK4Wy8csNV0uC51jQNRXUZvIa302XMtq5UlSTvbzFyNpO1OSD7U3UYNdPxSsRDqOno7adcmEvp7sETzI/lYecNzdPmGBweOeHZ3S9fwVwurX9PxdjstR1Kz0mxe81K4jtrZCoaWQ4VckAZP1IqyCGUFTkHkEd6ri2afTvs2p+TcmSMpPtiKxyZGD8pLYB9CTWD4Flkj0m90qWRpP7IvpbKN2OSY1w0YJ9kZR+FHVoOiZu2Gp2WqQyS6dcx3CRytC5Q52upwyn0INWq5SRRpHxOtvI+WHXLSTz4x0M0O0q/1KMQfoK6ujomHUKKKKQwooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAmt/4vwqaobf8Ai/Cpql7jQVUf/WN9TVuqj/6xvqaEDEoooqhBRRRQAUUUUAFFFFABRRRQAUUUUAFctdJeP8RcWE8ED/2UNzTQGUEeb6B1/nXU0VUZcruJq6sYU3hlJPDN7pYum829LPPcsgJaRiCW2jHHGAOwxW4o2qB6DFLRQ5N7hZGFfaLqd3rVrqCalaRmzMnkobJm4cYO4+aMnHcYq3NpTXGtafqMk4DWcciMix8OXAGevHTpzU02oeTqlvZfZLqTz1Y+fHHmKPA6M2eCe1XKfNJJCsrsz9P0v7BqGo3XneZ9umWXbtxswgXGc89M9qr6bpF3pUzx2t7CbB5nm8mS3JkXcckBw4GMknlTWxVeyvrfUbcz2cnmRh2QttI+ZTgjn3FHNILI5/xMJdI1nT/EsUUs0Fqj219HEpZhA5B8wKOTsZQSPQse1dOCGUEcgjIrN13Wk0WxSQQtdXVxIILW1jIDTyHouT0HUk9gCe1Xbi4FrZy3EqkiKMuyryeBkgZxUXSjd7L/AIf/AIPzKtqNvrb7bp9xa79nnRNHuxnbkYzisbUvCiaha6ZGt0YZbEIplEefNRcEqRnjJUHrxWpa6rb3WipqmHit2i84+YvzKuM8gZ7elMTW9Ok0qPUkuQ1pKwVJArckttAxjPXioag3r5f8AW6IbnR7g6tLf6ferbPcRLFMHh8zoThl+YYPJ6gj2qra+F/s2n6Zai8ZxYXRuN7R8yZLHB54+919uldBRRyRvf8Are/5huYFl4Z+xz2En2vf9jlnkx5WN/m9uvGM/j7VHD4YuE0v+ypdRRtOMjMyJblZWUsW2ly5GMnnC9PSujqnpepQ6tp6XdsrrGzMoEgAPysVPQnuKXs4XS/rp/wBle70mR9Si1DTrlba5SLyWEkXmRvHnIBUFTkHkEH86deadc6joV1YXt1EZLhGTzYYCoUEf3Sxz+daVFU4RaafUNncjgi8i2jizu8tAucYzgYqSiirbu7iWhm6hpclzf29/ZXItruBWTc8fmI6N1VlyD1AIIIp2m6YbOe5urib7Rd3TAyyBNi4UYVVXJwAPUk+9aFFQopO4zP0vTP7NN4fO837VdPcfdxt3Y4689OtaFZi60s19NbWdldXQt3Ec0sewIjdx8zAnA64BrToja1o7B1M/VdM/tMWg87yvs10lx93O7bnjrx161dliSeF4pRuSRSrD1B4NPop8q18w63MBfDtw1hFpl1qIm02JhiPyMSuinKoz7sEcDOFBIFSXfhuO81176Wc+RJB5UltsGGbDKG3f7rsMYrbrN1fWF0aHz57O5lgGN8sWzCEnABBYHqR0BqJQhb3hbEOkaJPpGitZxX5kuGkMhuWiHJyOq55GAAeQfcU600aRNabVb6eGW58nyV8iDyl25z82WYseB349K1qKrlWnkFjPbT7htfS/F/KtusBjNmB8jNnO7r1/D8a0KKKaSWwxkkgiieRgxVFLEKpY8egHJPsK5fwbb6ut/rt5qFxeSafd3nmafHe7hIiY+b5G5RcnAUgHAzgZ56uin1uJhVPUbe7ubUxWVxDAXBV2mgMmQRjgB1wfzq5UN3crZ2U1zIGZIY2kYL1IAzxQ2oq7GvIzdK0Waz0P+yb+5hu7ZYBAvlwGJtuMHcS7ZyPTFO0/TtUsI4LYalbz2kOFUS2h80oOgLiQDOOM7fwq9YXiahp8F5CrrHPGJFDjDAEZ5qxVucm231IUVZWM+20v7Prt9qPnbvtkcSeXtxs2buc55zu9Kq2vhyKHwnLoVxMZopo5Y3kC7TiQsTxk9N1bVRXMzwW7SRW8lyw6RRFQzfTcQP1qW3Jcr6lrR3Rh+Fbqe60VtL1eF0vtPAtbkOp2ygDCyKe6svPscg9KzpPA95LoMOiPrmNOtJI2tUS1w4VJAyrI2/58AYGAvqc9K6TSdUi1iwF1BFLEu94ykoG4FWKnoSOo9avU3fm13JVrWRz+oeHbyTxSuu6VqEFtcG0+ySLc2nnjZv3AqQ6FTnOeoPHHFT+HtA/sLTLmzlvHvvtFzNO0siBGPmNkg7eM89QAPYVs0VPS39dyut/67GD4d0PUNAtoNPGpW9zplshjgja0Kzhc/KGkEm04HGdgzWZL4IvTY61p1prUdvYatLNNJixDTq0g5HmF9rLn/YzjgMDgje07Wl1TUr6C1gY21m4iN0WG2SX+JFHfbwCfUkdjVjVNQi0nSbvULlXaG1haZ1jALEKMnGSOeKHtdjV72RWlsNSisbSHStRhge3jEbfaLUyxyAADO0OrA8cfNjk5B4IoWPg60g0HVNNu3WUavJJLdmCPyU3OoU7FydvAHUk55JNb1tOt1axXEYISVA6huoBGealpu93clWsrGPp1jr1pJDHd6vZ3drGNpzYMkzjHGXEu3d0yQgB54FN1bQ7i71iz1bS72O0vrWN4cz25mjkjfBIKhlOcqpBDevWtDTtStNVtTc2EvmxCR4i20r8yMVYYIHQgirVHmHkZt/qf9h6C97qjfaJIUAYW0JHnSE4ComWOWYgAZPXrVTwjpVzpmiM2pBVv72eS8ulU5CPI2dgPfaMLn2rdooA5XTQ+v8AjWXWlRl0/TYZLK0ZhjzpSw81x/sjYEB7kNXVUVVuNStLW/tLKeXZcXhcQJtJ37RluQMDA9aXRIOty1RUF7dJY2FxdzBjHbxNKwUckKMnHvxVPRNYbW7Fbsabd2UMiLJEbkxHzVYZBAR2xxjrjrQM06KpNq1ksxjMxyr+WX2NsDem/G3P40mq6jLptr58WnXV+Bkuts0QKADOT5jqMfTJqI1ISTcXew+VrdF6iqGm6pHq2g2+qWUMhjuYBPFE+FcgjIB5wD+OKi0DW4te037QkT280cjQ3NtIRvglX7yHH5g9wQe9aWd2ib9TUorJfXUg8Tx6Nd28kBuIjJaXDMClwy/fQdwwGDg9Rn0rWpDCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigCa3/AIvwqaobf+L8KmqXuNBVR/8AWN9TVuqj/wCsb6mhAxKKKKoQUUUUAFFFFABRRRQAUUUUAFFFFABXM3kc1948Fk97dRWY04SvDBM0YdvMIByCCPwIJwB04rpq5670OS98Z/bJ1nW0FgIhJDctEd/mZx8jBjx+FaQaT1Jl8P3fmZdzf3kPgnxAEupy9hcSQ29wXPmbQVx83UkZIz14rdttGngvLW7TVbtyob7Sk0hdbjI4wudqYPPygelWm0bT30dtLNsosmXaYlJXIzk8jnJPJOcmroGAAOgpuemguXuc7fI1p420fyLi5CXYuDNG1zI0bbUBHyE7RjPYCk1wXFx4r0eyjvbi2t5o5zMsEhXzAApAyOn1HI5wRV+48N6ddXi3c/2szqWKOL6ddm7rtw/yg+gxVttOtnu7a6eMtPaoyROzsSoYAHvznA5OaFJKz7J/r/mFnr/XYxtFRrmfWtLuJ7mS3tbpViY3MgkVSitjzAd2Mk9/aq3grTIBpf2jzLrel3OAPtcuw4kYcpu2n8Rz1rpLext7W4uZ4I9kl04eZtxO5gAAeenAHSq8WiWMF611bpNDI8hkdY7iRY2YjkmMNtJP0p+00t6feLl/NmRdYm+KOnpOMrBpc0sAJ43mRFYgeoXjPoxrpmVXUq4DKwwQRkEVheJNKvbiSz1XRBGdU05yY45W2rcRsMPEW7ZwCD2ZRW6CSoJGDjkelYr4bev+Zp1uZ2rxRweGb6KCNY40tJAqIuAo2ngAVxc+dM0qKwPFvfC1u4PZ98YkX+Tfia9CnhjubeSCZd0cqlHXOMgjBFU7rRNOvbW1t7m2EkdoytANzAoVGByDn86ylTblzLy/W/6fcJ6q3qVLO4mbxlqcLyuYUt4WWMsdqk7skDtWBb3l5deHtCxf3Ctc6i8Ukqyks6Zk4z9Bx6cY6V1d5o1jf3Hn3MTGTZ5bFJXTen91gpAYexyKSLQ9Oht7aCK2CRWkpmgUM2Ec556+546UuSTavt/9sn+WgPrb+tLHLaFd3g1LSWkvbmYXE11C6SSsy7E+6ME9Rj73XnrUVhDLa+ApdUivrqOeB5HgVJSqLiU/KUHDZPXcD1rroNE0+2kt3ht9rWzyPEd7HaX+8evOfeoovDelwsu2CQormRYXnkaIMTnIjLbepz04qPZS09P8tfvQ/wCvzM3Ur26uvEUFiYL54FtBcNDZzrE7MSV5YupwPQHqeajnk1Wz8G6ybr7TA0IkNo8sytKI8ZGWUnkHIznNb97pdpqDxyXMbebFny5Y5Gjdc9QGUg4PpnFIdKtX02WxlEs1vMCHWad3JB6jcxJ/WqlTk1Kz3T/Hb7gW6KFjpNwPsF6uqXbShd1wJZC6zgr02ZCpg9CB+dblNjRYo1jQYVQFA9AKdW9km7CSsjD1KSS68SWmltPLBbPbvO/lSGNpSGAC7hhgBnPBGaj00XA1XVtKjvbnyYBE8UzMJJIi4JK7nBz0754Nat9plpqPl/aoyWibdHIjsjoe+GUgj86dZWFtp8TR2kewOxZ2LFmdj3Zjkk+5NZKD5rsbOd8L2U/n6m/9p3W2PUpAybYsS4xy3yZ59iPbFdXVe1sbey877NHs8+UzSfMTuc9Tz9KsVVOPJBRF1ZjeJLqeGCytreZrc3t2lu8yY3IpBJwT0JxgH3qwbZdIsrq4tnuZtsRcRzXDy5KgnguSRn64q1eWdvqFq9veQrNE/VWH6j0PuORTLTTreyLmHzWZwAXmmeVsemXJIHtScXd+fXtp/TH1TOZuDcW3hGDXY9Qu5L/bHMSZmMchYgFPLzsA+bAwAeBzmtHxoc+Ebnt80X/oxauReHtMglR47dgsbmRIjK5iRuuVjJ2g8noOKk1HRbLVhi+EzpgAxrcyIhwcjKqwBOe+KTjKzsKxT8WzzW3h95LeV4n86Ibo2KnBkUHke1R6jdSp4xsbeOd1jaznd41cgEgjBI/PmtM6XaPp0ljMsk9vICGWeZ5Cf+BMSfpzx2qG30DTba8F3HAxuQpTzpJndyDgEFmJJHAxnp2pSjNvT+tB/wBficTaT38tjbPLql6xm0ua5cidhlkOExg8Y4yRycc5rZjkutT1bRYZb65iin0zzp1hkKeacr3HI57jB7Z5rdTw/pkcccaW2Fjga3UeY3EbdV6/r1qaHSrOCe3mih2yW8PkRHcTtTjjrz0HJ5pRpNb7f8P/AJoX9fl/kzJ0eS+m07WbS2uWM1tdSw2sk7FynAK5JyTgnvn8aj0S4+yX9tZ6lFqVrfzQn/j4uTPFOR1KncQDxnGF6/SttNLs44rqNYBsu3Mk6kk72IwTz06dqZbaPZ2tyLhFlkmVSqvPcSTFAeu3exxn2pxhJOL7L+v6+8GQ6jAZNY02QasbPy3c/ZQwH2rjpjPOPofwrSdFkjZJFDIwwysMgj0qtc6ZZ3l5bXdzAJJ7UloXJPyE9fr+NW60S3uPqMjjSGJYoUWONAFVFGAoHYCoor60nvJrSC6hkubfHnQpIC8eeRuUcjPbNT1zWh+GZbXxXqfiS/S3gvL+JITBauXRVX+IuVUsxwP4RgDHPWrW4umh01FFQ3VrHeQGKVpVUnOYpnib/vpSD+tAzC8IOY/DUzhdxW6uSAO/71qxrKbWb/RLfU7K21WXUZSswkN3GLdhnlPLMuAu3I+7uzz1rq9N0Ox0libAToDnKPdSuuSck7WYjJPfGaanh/T4pjJAtxBl95jgu5Y493rsVgvPfjmtlOKk3/X5mbi2rGZdxzX3jwWT3t1DaDThK8EEzRh28wgcqQR+BBOAOnFP0O9um8OalmR55bOe4ihZ2yzBCdoJPJPbJra+wW39pHUPL/0oxeT5m4/cznGOnWi0sLaxjlS1i2LNI0rjcTuZjknms5SvDl8ikrS5v62/zMfwHHHH4B0Uw4PmWccrkHO52G5yfcsST71y+qAa98P/ABHrWq3l2JI/tkMMUd5JDFEsbMiqURgrE4ydwJJbHTArq/Del3uhSXemMI30qN/MsJA3zRqxJMRHop6H0IHan3PhDRLya6kuLNnF2SZovPkETMV2l/LDbQ+P4gN3fOaifvXa6/gVH3bX6P7zNvr62XS9E0149Turm6tg8dpps/ku6qi5dpN6bQMgffGSw4PaDwiZtUg17SdRbU44bS/CQpcXp+0xIyJJsM0bljgscHeTggE9q377w3peoi0+0QyK9kpW3lguJIZI1IwQHRg2CAMjPOKl0rQ9N0T7R/Zdqtv9pcSTbWJ3vgDccnqQOT3PJ5JNU3eTff8AzJSaSXb/ACOW+HWiWqaL9sEt95sd9dAKdQnMZxM45jL7Tx3IOTyeeal1+6h1PUNRtbODxDeT2UYSSTTL0W0dq7LkDmWMO2MNyHxkD2rfg8Oaba6m99apcQSySGV0iu5UhdyMFjEG2EnvleTz1qO68J6NeahPeT2rmS5AFwi3EixXGBgeZGGCPxx8wPFS7tFLR3MNLnX9f+F+k3unO0t/PDDLcqkwge4THzqrgYRm9RjHqKsaVrVvaafrUNpbarFe6bEZ3sdUmaZ/ukqVkLvuVtpHDkAg8Dvtf8I9pq6Pb6XHA8VragCARTujxYGBtkVg4OCRkHoSOhqTTtGsdKWX7JG5ecgyyzzPNJJgYAZ3JYgDoCcCnLXmt1FHS1+hw2ltr9xp2kalp9lrs19M8M1zc3N/AbWeJyDJiITFVG0krtQEYHvm/wCJtEtbjx94daSW/BuXud/l6hOgGIuNgVxs99uM966C18K6XZXCS2S3dsqPvSCG/nSBTnOBEH2AZ7bce1W9T0ey1iKNL+JmML+ZFJHK0UkbdMq6EMpwSOD0OKHboKz6lTVrSOx8GalbwNMyJZzYM87zPyrHl3JY/iaXwxu/4QvSdn3vsEOPr5Yp99pmzwte6dYedKz20qRie4aV2ZlOAXkYk8nueKh8KNeR+H7Oy1DTLmwltLeKE+e8TCQhQCV8t2447460t+Zen6je8fn+hmxmS48GtvzFbxQsWJ+9NJnj6Ddj3J9uu/cb/wDhH5fN+/8AZTuz67eacdLszMZDEcl/MKb22FvXZnbn3xUWtz3MOmyJZ6bc6hJMrR7LdolK5B5PmOox9Mn2rgoYedKL5tW0l91/u326d3c6JTUpK3e5Q8Cf8k+0H/rwh/8AQBVbTf3HxO1uGH/VTWNtPKB0EmXTP1KqPyq/4Xs7rSfBOmWl1ARd2tkiPCHUneq/dznHXjOcUzwzpN1ZJd6jq5U6pqUgmuFQ5WEAYSJT3CjjPcknvXpT1qN+v9fr8jlj8CRT8ffutEsrqPi5ttTtXg9SxlVCPxVmH411NYGoaXeax4os2ulEWk6aRcRjcCbmfBAyOyoOfckdhzv1C2+f+X+RT3/rzCiiigYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBNb/xfhU1Q2/8AF+FTVL3Ggqo/+sb6mrdVH/1jfU0IGJRRRVCCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigCa3/AIvwqaobf+L8KmqXuNBVdonLkgd/WrFFAyt5L/3f1o8l/wC7+tWaKLisVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsVvJf+7+tHkv/AHf1qzRRcLFbyX/u/rR5L/3f1qzRRcLFbyX/ALv60eS/939as0UXCxW8l/7v60eS/wDd/WrNFFwsRQoyZ3DGaloopDP/2Q==)

The explore\_data() function:

* Takes in four parameters:
  + dataset, which is expected to be a list of lists.
  + start and end, which are both expected to be integers and represent the starting and the ending indices of a slice from the data set.
  + rows\_and\_columns, which is expected to be a Boolean and has False as a default argument.
* Slices the data set using dataset[start:end].
* Loops through the slice, and for each iteration, prints a row and adds a new line after that row using print('\n').
  + The \n in print('\n') is a special character and won't be printed. Instead, the \n character adds a new line, and we use print('\n') to add some blank space between rows.
* Prints the number of rows and columns if rows\_and\_columns is True.
  + dataset shouldn't have a header row, otherwise the function will print the wrong number of rows (one more row compared to the actual length).

To help you better understand what print('\n') does, we printed three rows from the AppleStore.csv data set below. In the first code cell, we don't use print('\n') between rows, while in the second one we do:



Now let's open the two data sets and explore them.

If you get stuck during the following exercise, you can check the [solution notebook](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb).

Instructions

1. Open the two data sets we mentioned above, and save both as lists of lists.
   * The App Store data set is stored in a CSV file named AppleStore.csv, and the Google Play data set is stored in a CSV file named googleplaystore.csv.
   * Both CSV files can be opened directly in the Jupyter Notebook interface you see on the right of the screen.
   * If you run into an error named UnicodeDecodeError, add encoding="utf8" to the open() function (for instance, use open('AppleStore.csv', encoding='utf8')).
2. Explore both data sets using the explore\_data() function.
   * Print the first few rows of each data set.
   * Find the number of rows and columns of each data set (recall that the function assumes the argument for the dataset parameter doesn't have a header row).
3. Print the column names and try to identify the columns that could help us with our analysis. Use the documentation of the data sets if you're having trouble understanding what a column describes. Add a link to the documentation for readers if you think the column names are not descriptive enough.

In the previous step, we opened the two data sets and performed a brief exploration of the data. Before beginning our analysis, we need to make sure the data we analyze is accurate, otherwise the results of our analysis will be wrong. This means that we need to:

* Detect inaccurate data, and correct or remove it.
* Detect duplicate data, and remove the duplicates.

Recall that at our company, we only build apps that are *free* to download and install, and that are directed toward an *English-speaking* audience. This means that we'll need to:

* Remove non-English apps like *爱奇艺PPS -《欢乐颂2》电视剧热播*.
* Remove apps that aren't free.

This process of preparing our data for analysis is called **data cleaning**. Data cleaning is done *before* the analysis; it includes removing or correcting wrong data, removing duplicate data, and modifying the data to fit the purpose of our analysis.

It's often said that data scientists spend around 80% of their time cleaning data, and only about 20% actually analyzing (cleaned) data. In this project, we'll see that this is not far from the truth.

Let's begin by detecting and deleting wrong data. For this guided project, we'll guide you throughout the entire data cleaning process. In later courses, we'll learn more about data cleaning, and you'll be able to perform data cleaning without any guidance.

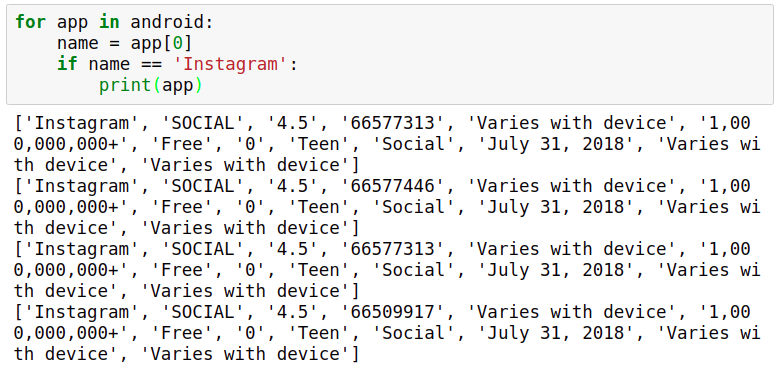
If you get stuck during the following exercise, you can check the [solution notebook](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb).

Instructions

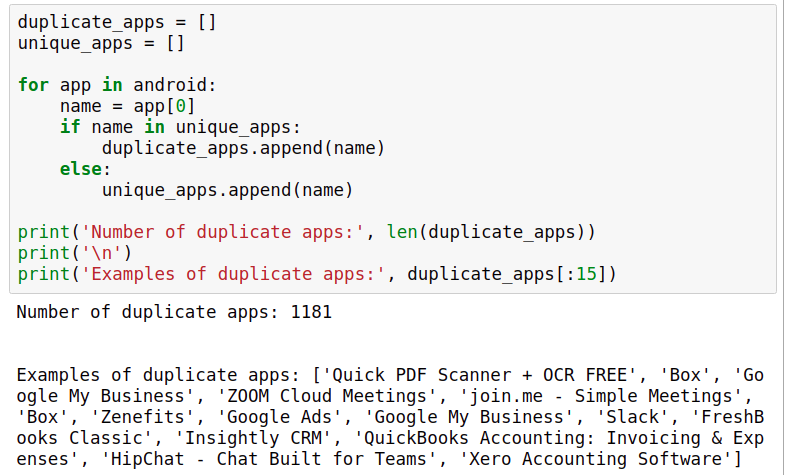
The Google Play data set has a dedicated [discussion section](https://www.kaggle.com/lava18/google-play-store-apps/discussion), and we can see that [one of the discussions](https://www.kaggle.com/lava18/google-play-store-apps/discussion/66015) describes an error for a certain row.

1. Read the discussion and find out what the index of the row is.
2. Print the row at that index to check whether it's indeed incorrect. Take into account the user reporting the error might or might have not removed the header row, so the index number might vary.
3. If the row has an error, remove the row using the [del statement](https://docs.python.org/3/reference/simple_stmts.html?highlight=del#the-del-statement). For instance, to remove the row with the index 149 from a data set data that is stored as a list of list, you can use the code del data[149].
4. Make sure you don't run the del statement more than once, otherwise you'll delete more than one row.
5. Read the [discussion section](https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps/discussion) for the App Store data set, and see whether you can find any reports of wrong data.

In the last step, we started the data cleaning process and deleted a row with incorrect data from the Google Play data set. If you explore the Google Play data set long enough or look at the [discussions](https://www.kaggle.com/lava18/google-play-store-apps/discussion) section, you'll notice some apps have duplicate entries. For instance, Instagram has four entries:



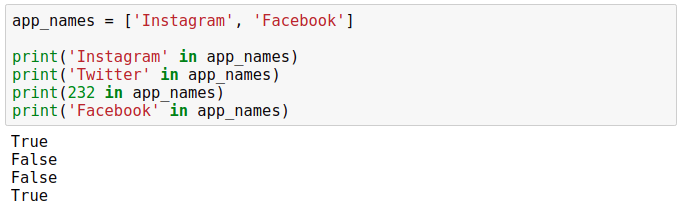
In total, there are 1,181 cases where an app occurs more than once:



Above, we:

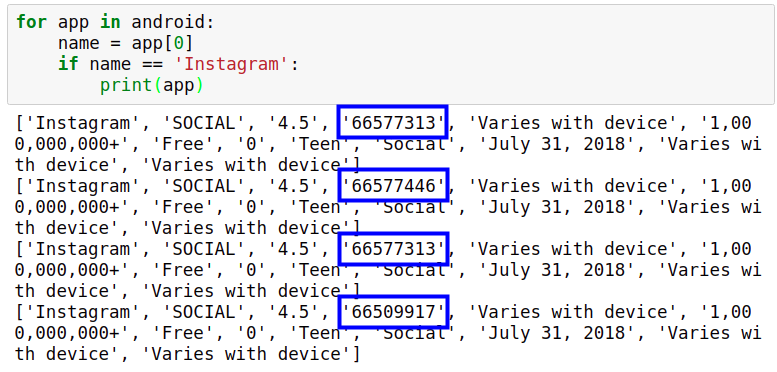
* Created two lists: one for storing the name of duplicate apps, and one for storing the name of unique apps.
* Looped through the android data set (the Google Play data set), and for each iteration:
  + We saved the app name to a variable named name.
  + If name was already in the unique\_apps list, we appended name to the duplicate\_apps list.
  + Else (if name wasn't already in the unique\_apps list), we appended name to the unique\_apps list.

(As a side note, you may notice we used the in operator above to check for membership in a list. We only learned to use in to check for membership in dictionaries, but in also works with lists):



We don't want to count certain apps more than once when we analyze data, so we need to remove the duplicate entries and keep only one entry per app. One thing we could do is remove the duplicate rows randomly, but we could probably find a better way.

If you examine the rows we printed for the Instagram app, the main difference happens on the fourth position of each row, which corresponds to the number of reviews. The different numbers show the data was collected at different times.



We can use this information to build a criterion for removing the duplicates. The higher the number of reviews, the more recent the data should be. Rather than removing duplicates randomly, we'll only keep the row with the highest number of reviews and remove the other entries for any given app.

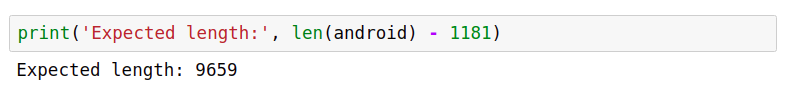
We'll remove the rows on the next screen. Now it's your turn to write some code and confirm the data has duplicate entries.

If you get stuck during the following exercise, you can check the [solution notebook](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb).

Instructions

1. Using a combination of narrative and code, explain the reader that the Google Play data set has duplicate entries. Print a few duplicate rows to confirm.
2. Count the number of duplicates using the technique we learned above.
3. Explain that you won't remove the duplicates randomly. Describe the criterion you're going to use to remove the duplicates.
   * We already suggested a criterion above, but you can come up with another criterion if you want. Make sure you support your criterion with at least one argument.

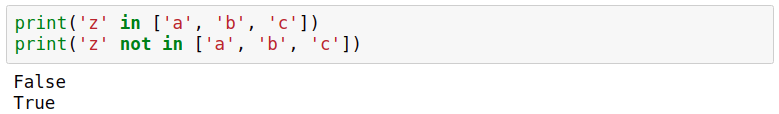
On the previous screen, we looped through the Google Play data set and found that there are 1,181 duplicates. After we remove the duplicates, we should be left with 9,659 rows:



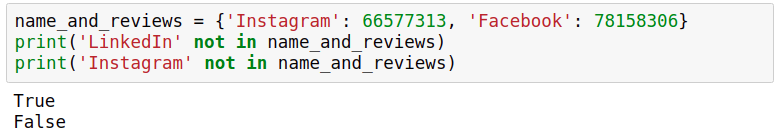
To remove the duplicates, we will:

* Create a dictionary, where each dictionary key is a unique app name and the corresponding dictionary value is the highest number of reviews of that app.
* Use the information stored in the dictionary and create a new data set, which will have only one entry per app (and for each app, we'll only select the entry with the highest number of reviews).

To turn the steps above into code, we'll need to use the [not in operator](https://docs.python.org/3/reference/expressions.html#not-in). The not in operator is the opposite of the in operator. For instance, 'z' in ['a', 'b', 'c'] returns False because 'z' is not in ['a', 'b', 'c'], but 'z' not in ['a', 'b', 'c'] returns True because it's true that 'z' is *not in* the list ['a', 'b', 'c'].



Essentially, we use both the in and not in operators to check for membership — we want to know whether a value belongs to some group of values or not. We can also use the not in operator to check for membership in a dictionary. Just like in the case of the in operator, the membership check is only done over the dictionary keys:



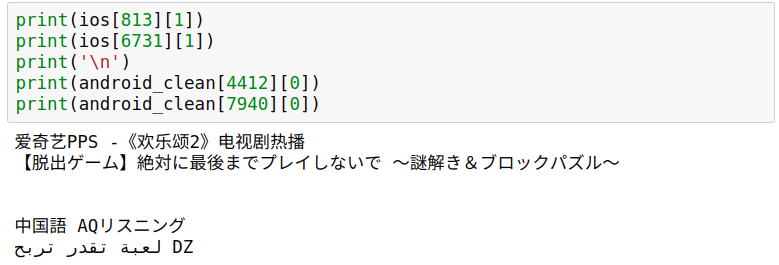
Now let's write the code to remove the duplicate entries.

If you get stuck during the following exercise, you can check the [solution notebook](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb).

Instructions

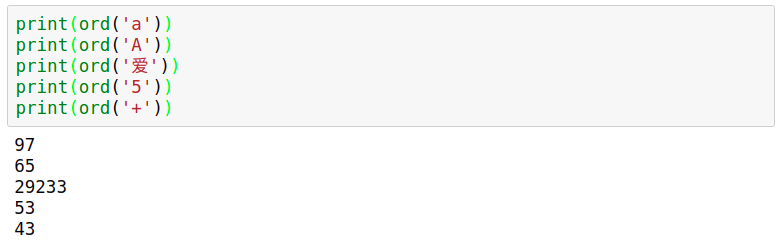
1. Create a dictionary where each key is a unique app name and the corresponding dictionary value is the highest number of reviews of that app.
   * Start by creating an empty dictionary named reviews\_max.
   * Loop through the Google Play data set (make sure you don't include the header row). For each iteration:
     + Assign the app name to a variable named name.
     + Convert the number of reviews to float. Assign it to a variable named n\_reviews.
     + If name already exists as a key in the reviews\_max dictionary **and** reviews\_max[name] < n\_reviews, update the number of reviews for that entry in the reviews\_max dictionary.
     + If name **is not in** the reviews\_max dictionary as a key, create a new entry in the dictionary where the key is the app name, and the value is the number of reviews. Make sure you don't use an else clause here, otherwise the number of reviews will be incorrectly updated whenever reviews\_max[name] < n\_reviews evaluates to False.
   * Inspect the dictionary to make sure everything went as expected. Measure the length of the dictionary — remember that the expected length is 9,659 entries.
2. Use the dictionary you created above to remove the duplicate rows:
   * Start by creating two empty lists: android\_clean (which will store our new cleaned data set) and already\_added (which will just store app names).
   * Loop through the Google Play data set (make sure you don't include the header row), and for each iteration:
     + Assign the app name to a variable named name.
     + Convert the number of reviews to float, and assign it to a variable named n\_reviews.
   * If n\_reviews is the same as the number of maximum reviews of the app name (the number can be found in the reviews\_max dictionary) **and** name is not already in the list already\_added (read the solution notebook to find out why we need this supplementary condition):
     + Append the entire row to the android\_clean list (which will eventually be a list of list and store our cleaned data set).
     + Append the name of the app name to the already\_added list — this helps us to keep track of apps that we already added.
3. Explore the android\_clean data set to ensure everything went as expected. The data set should have 9,659 rows. The two steps above are a bit more involved, so make sure you use Markdown to explain the readers the steps you took.

In the previous step, we managed to remove the duplicate app entries in the Google Play data set. Remember we use English for the apps we develop at our company, and we'd like to analyze only the apps that are directed toward an English-speaking audience. However, if we explore the data long enough, we'll find that both data sets have apps with names that suggest they are not directed toward an English-speaking audience.



We're not interested in keeping these apps, so we'll remove them. One way to go about this is to remove each app with a name containing a symbol that is not commonly used in English text — English text usually includes letters from the English alphabet, numbers composed of digits from 0 to 9, punctuation marks (., !, ?, ;), and other symbols (+, \*, /).

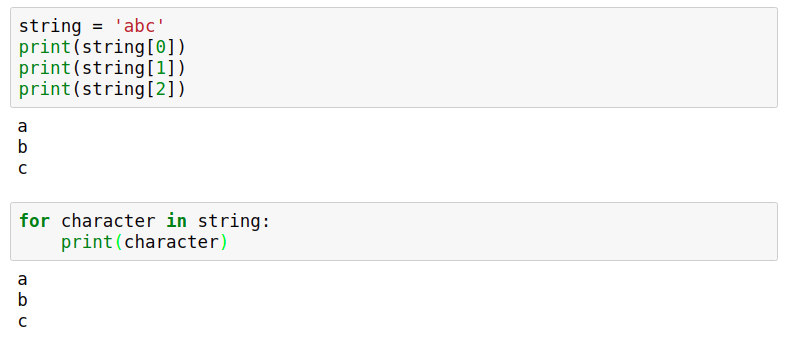
Behind the scenes, each character we use in a string has a corresponding number associated with it. For instance, the corresponding number for character 'a' is 97, character 'A' is 65, and character '爱' is 29,233. We can get the corresponding number of each character using the [ord() built-in function](https://docs.python.org/3/library/functions.html" \l "ord" \t "_blank).



The numbers corresponding to the characters we commonly use in an English text are all in the range 0 to 127, according to the [ASCII](https://en.wikipedia.org/wiki/ASCII) (American Standard Code for Information Interchange) system. Based on this number range, we can build a function that detects whether a character belongs to the set of common English characters or not. If the number is equal to or less than 127, then the character belongs to the set of common English characters.

If an app name contains a character that is greater than 127, then it probably means that the app has a non-English name. Our app names, however, are stored as strings, so how could we take each individual character of a string and check its corresponding number?

In Python, strings are indexable and iterable, which means we can use indexing to select an individual character, and we can also iterate on the string using a for loop.



Let's first try to write the function we talked about above, and in the next screen we'll remove the rows corresponding to the non-English apps.

If you get stuck during the following exercise, you can check the [solution notebook](https://github.com/dataquestio/solutions/blob/master/Mission350Solutions.ipynb).

Instructions

1. Write a function that takes in a string and returns False if there's any character in the string that doesn't belong to the set of common English characters, otherwise it returns True.
   * Inside the function, iterate over the input string. For each iteration check whether the number associated with the character is greater than 127. When a character is greater than 127, the function should immediately return False — the app name is probably non-English since it contains a character that doesn't belong to the set of common English characters.
   * If the loop finishes running without the return statement being executed, then it means no character had a corresponding number over 127 — the app name is probably English, so the functions should return True.
2. Use your function to check whether these app names are detected as English or non-English:
   * 'Instagram'
   * '爱奇艺PPS -《欢乐颂2》电视剧热播'
   * 'Docs To Go™ Free Office Suite'
   * 'Instachat 😜'